Ch. 9 – BGP (Part 2)

Cabrillo College

CCNP 1 version 3.0

Rick Graziani

Cabrillo College

Note to instructors

- If you have downloaded this presentation from the Cisco Networking Academy Community FTP Center, this may not be my latest version of this PowerPoint.
- For the latest PowerPoints for all my CCNA, CCNP, and Wireless classes, please go to my web site:

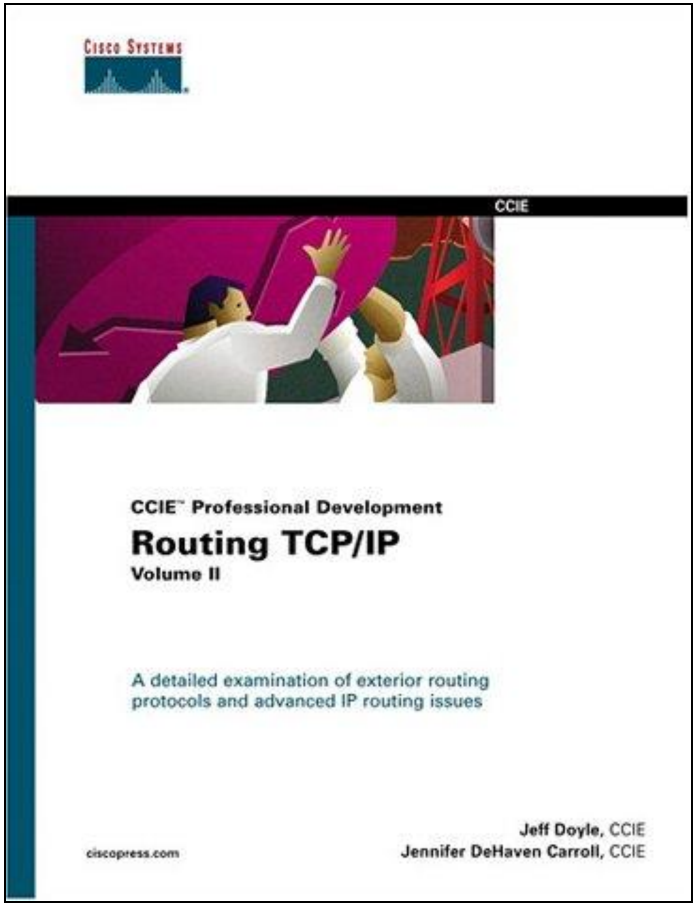
  http://www.cabrillo.cc.ca.us/~rgraziani/

  - The username is *cisco* and the password is *perlman* for all of my materials.
- If you have any questions on any of my materials or the curriculum, please feel free to email me at graziani@cabrillo.edu (I really don't mind helping.) Also, if you run across any typos or errors in my presentations, please let me know.
- I will add "(Updated – *date*)" next to each presentation on my web site that has been updated since these have been uploaded to the FTP center.

Thanks! Rick

Concepts, diagrams, and examples

This presentation is based partly on information from the books *Routing TCP/IP Vol. II by Jeff Doyle and Jennifer Carroll* and *Cisco BGP Command and Configuration Handbook by Parkhurst.*



Cisco Systems

CCIE

CCIE™ Professional Development

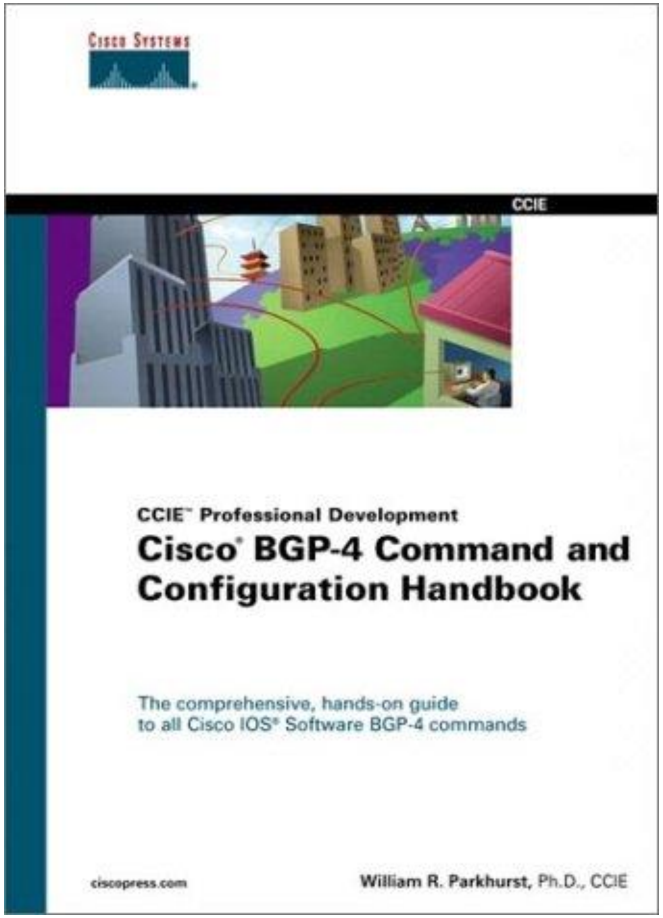
Routing TCP/IP

Volume II

A detailed examination of exterior routing protocols and advanced IP routing issues

ciscopress.com

Jeff Doyle, CCIE
Jennifer DeHaven Carroll, CCIE



Cisco Systems

CCIE

CCIE™ Professional Development

Cisco® BGP-4 Command and Configuration Handbook

The comprehensive, hands-on guide to all Cisco IOS® Software BGP-4 commands

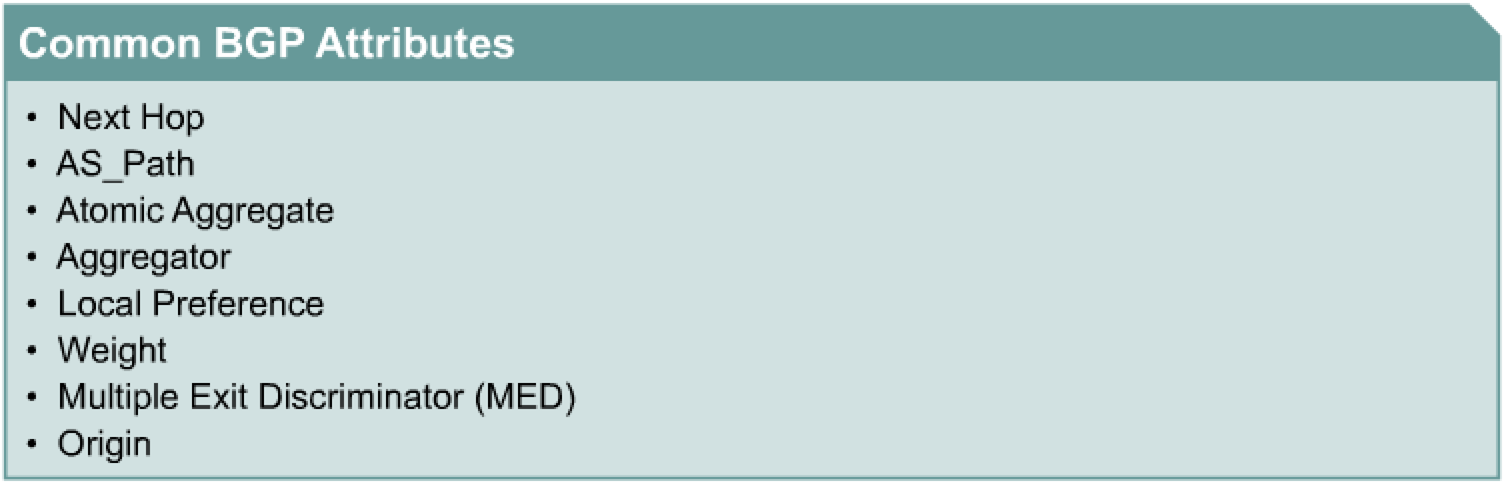
ciscopress.com

William R. Parkhurst, Ph.D., CCIE

Other source, Cisco on-line

- Quite a few of the examples in this presentation are taken from Cisco's web site:
  - http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/icsbgp4.htm
- At the time of creating this presentation I did not have time to incorporate the graphics from CCNP 1 version 3.0. I hope to do this in the near future.

Implementing Policy



Common BGP Attributes

- Next Hop
- AS_Path
- Atomic Aggregate
- Aggregator
- Local Preference
- Weight
- Multiple Exit Discriminator (MED)
- Origin

- Traffic inside and outside an AS always flows according to the road map laid out by routes.
- Altering the routes changes traffic behavior.
  - How do I prevent my private networks from being advertised?
  - How do I filter routing updates coming from a particular neighbor?
  - How do I make sure that I use this link or this provider rather than another one?

Using BGP Attributes

- When a BGP speaker receives updates from multiple autonomous systems that describe different paths to the same destination, it must choose the single best path for reaching that destination.
  - Once chosen, BGP propagates the best path to its neighbors.
  - The decision is based on the value of attributes (such as NEXT_HOP or LOCAL_PREF) that the update contains and other configurable BGP factors.

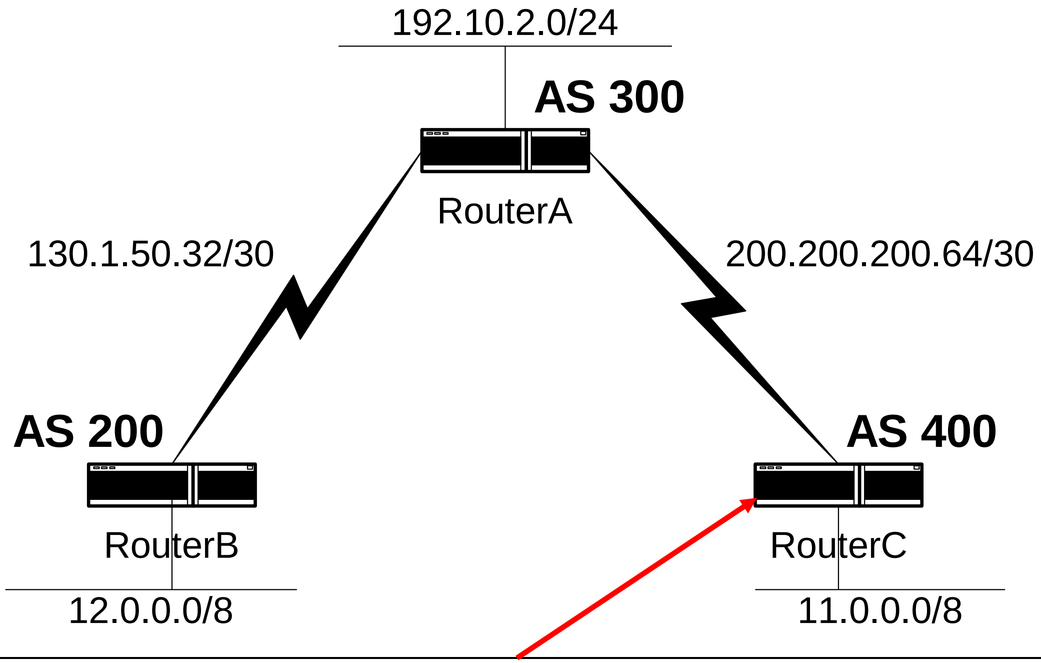
show ip bgp

To display entries in the BGP routing table, use the **show ip bgp EXEC command.**

```
show ip bgp [network] [network-mask] [longer-prefixes]
```

- Let’s look at an example, but some of the options will be discussed later.

show ip bgp



```
RouterC#show ip bgp
BGP table version is 8, local router ID is 200.200.200.66
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         200.200.200.65                         0 300 200 i
*> 192.10.2.0       200.200.200.65           0             0 300 i
```

show ip bgp

```
RouterC#show ip bgp
BGP table version is 8, local router ID is 200.200.200.66
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         200.200.200.65                         0 300 200 i
*> 192.10.2.0       200.200.200.65           0             0 300 i
```

- **BGP table version** - Internal version number of the table. This number is incremented whenever the table changes.
- **local router ID** - IP address of the router.
- **Status codes** - Status of the table entry. The status is displayed at the beginning of each line in the table. It can be one of the following values:

  s —The table entry is suppressed.

  * —The table entry is valid.

  > —The table entry is the best entry to use for that network.

  i —The table entry was learned via an internal BGP (iBGP) session

```
RouterC#show ip bgp
BGP table version is 8, local router ID is 200.200.200.66
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         200.200.200.65                         0 300 200 i
*> 193.10.2.0       200.200.200.65           0             0 300 i
```

- **Origin codes** - Origin of the entry. The origin code is placed at the end of each line in the table. It can be one of the following values:
  - i —Entry originated from Interior Gateway Protocol (IGP) and was advertised with a **network** router configuration command.
  - e —Entry originated from Exterior Gateway Protocol (EGP).
  - ? —Origin of the path is not clear. Usually, this is a router that is redistributed into BGP from an IGP.
- **Network** - IP address of a network entity.
- **Next Hop** - IP address of the next system that is used when forwarding a packet to the destination network. An entry of 0.0.0.0 indicates that the router has some non-BGP routes to this network

```
RouterC#show ip bgp
BGP table version is 8, local router ID is 200.200.200.66
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         200.200.200.65                         0 300 200 i
*> 193.10.2.0       200.200.200.65           0             0 300 i
```

- **Metric** - If shown, the value of the interautonomous system metric.
- **LocPrf** - Local preference value as set with the **set local-preference** route-map configuration command. The default value is 100.
- **Weight** - Weight of the route as set via autonomous system filters.
- **Path** - Autonomous system paths to the destination network. There can be one entry in this field for each autonomous system in the path.

BGP Attributes

- ORIGIN
- NEXT_HOP
- AS_PATH
- LOCAL_PREF
- Weight
- MULTI_EXIT_DISC (MED)
- ATOMIC_AGGREGATE
- **NOTE**: For several of these attributes, multiple options have been included in this presentation. Because of time and to make sure we first grasp the basic concepts, some of the options are added to the presentation for your own information and reference.

Summary of the BGP Path Selection Process

- BGP selects only one path as the best path.
- When the path is selected, BGP puts the selected path in its routing table and propagates the path to its neighbors.
- BGP uses the following criteria, in the order presented, to select a path for a destination:

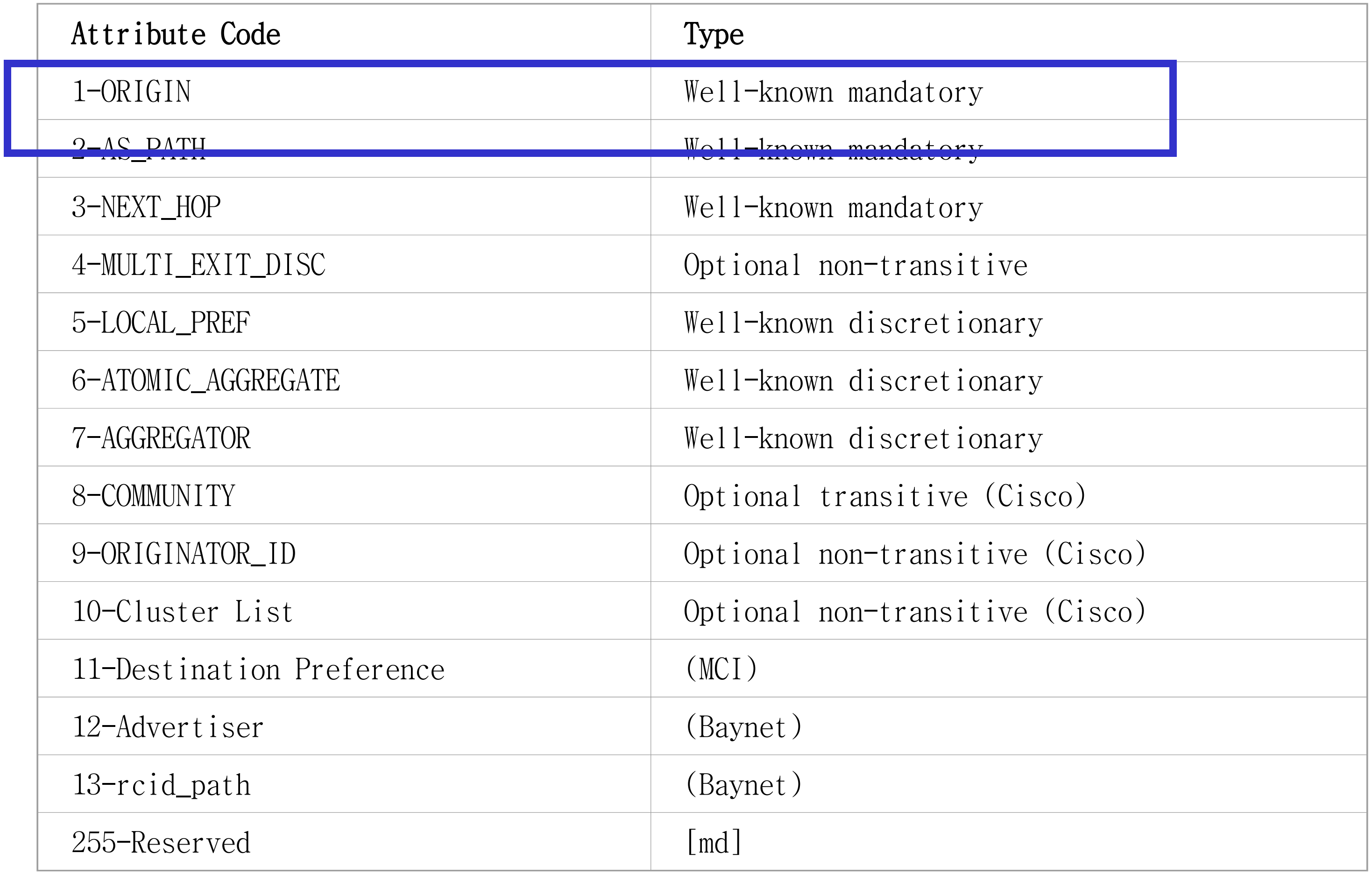
1. If the path specifies a next hop that is inaccessible, drop the update.
2. Prefer the path with the **largest weight**.
3. If the weights are the same, prefer the path with the **largest local preference**.
4. If the local preferences are the same, prefer the **path that was originated by BGP** running on this router.
5. If no route was originated, prefer the route that has the **shortest AS_path**.
6. If all paths have the same AS_path length, prefer the path with the **lowest origin** type (where IGP is lower than EGP, and EGP is lower than Incomplete).
7. If the origin codes are the same, prefer the path with the **lowest MED attribute**.
8. If the paths have the same MED, prefer the **external path** over the internal path.
9. If the paths are still the same, prefer the path through the **closest IGP neighbor**.
10. Prefer the path with the lowest IP address, as specified by the BGP **router ID**.

BGP Best Path Selection Algorithm

More info:

- http://www.cisco.com/warp/public/459/25.shtml

Path Attributes



Attribute Code	Type
1-ORIGIN	Well-known mandatory
2-AS_PATH	Well-known mandatory
3-NEXT_HOP	Well-known mandatory
4-MULTI_EXIT_DISC	Optional non-transitive
5-LOCAL_PREF	Well-known discretionary
6-ATOMIC_AGGREGATE	Well-known discretionary
7-AGGREGATOR	Well-known discretionary
8-COMMUNITY	Optional transitive (Cisco)
9-ORIGINATOR_ID	Optional non-transitive (Cisco)
10-Cluster List	Optional non-transitive (Cisco)
11-Destination Preference	(MCI)
12-Advertiser	(Baynet)
13-rcid_path	(Baynet)
255-Reserved	[md]

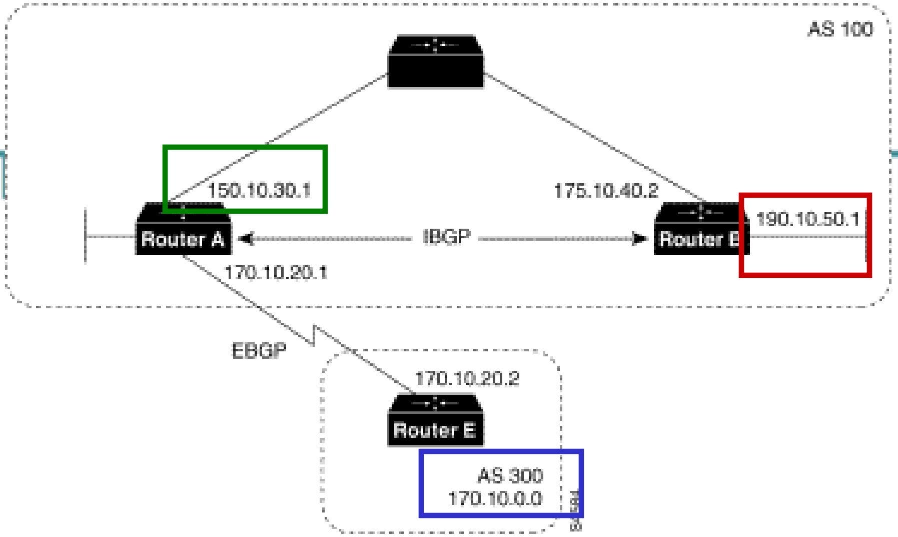
The ORIGIN attribute

- Well-known mandatory attribute (type code 1)
- Indicates the origin of the routing update
  - **IGP:** The prefix is internal to the originating AS.
  - **EGP**: The prefix was learned via some EGP, such as BGP.
  - **INCOMPLETE**: The prefix was learned by some other means, probably redistribution.
- BGP considers the ORIGIN attribute in its decision-making process to establish a preference ranking among multiple routes.
- Specifically, BGP prefers the path with the lowest origin type, where
  - IGP is lower than EGP
  - and EGP is lower than INCOMPLETE.

The ORIGIN attribute

The origin attribute provides information about the origin of the route. The origin of a route can be one of three values:

- ***IGP***—The route is interior to the originating AS.
  - This value is set when the network router configuration command is used to inject the route into BGP.
  - The IGP origin type is represented by the letter **i** in the output of the **show ip bgp** EXEC command.
- ***EGP***—The route is learned via the Exterior Gateway Protocol (EGP).
  - The EGP origin type is represented by the letter **e** in the output of the **show ip bgp** EXEC command.
- ***Incomplete***—The origin of the route is unknown or learned in some other way.
  - An origin of Incomplete occurs when a route is **redistributed** into BGP.
  - The Incomplete origin type is represented by the **?** symbol in the output of the **show ip bgp** EXEC command



```
Router A
router bgp 100
   neighbor 190.10.50.1 remote-as 100
   neighbor 170.10.20.2 remote-as 300
   network 150.10.0.0
   redistribute static
ip route 190.10.0.0 255.255.0.0 null 0

Router B
router bgp 100
   neighbor 150.10.30.1 remote-as 100
   network 190.10.50.0

Router E
router bgp 300
   neighbor 170.10.20.1 remote-as 100
   network 170.10.0.0
```

Given these configurations, the following is true:

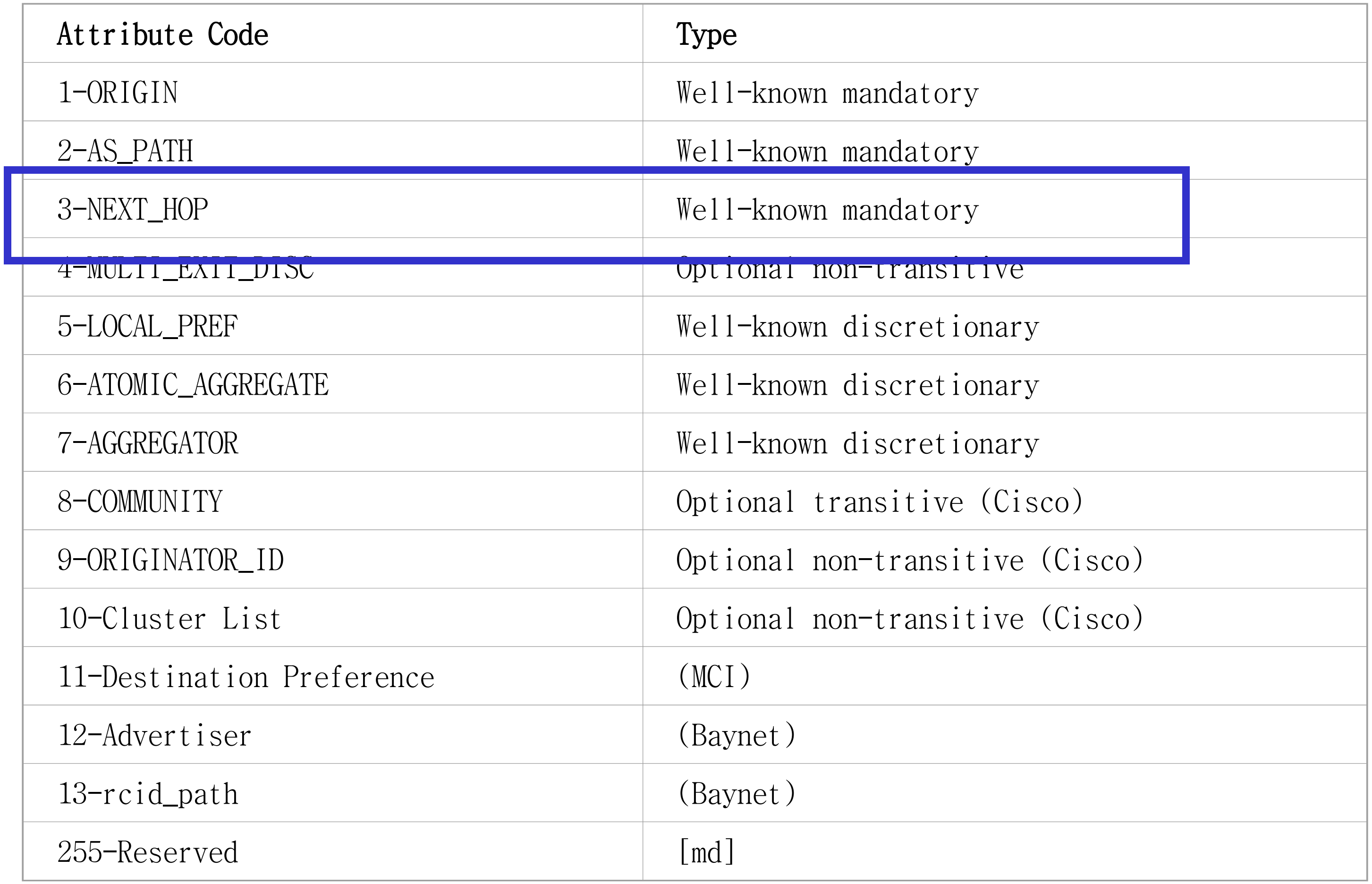
- **From Router A**, the route for reaching **170.10.0.0** has an AS_path of 300 and an origin attribute of **IGP**.
- **From Router A**, the route for reaching **190.10.50.0** has an empty AS_path (the route is in the same AS as Router A) and an origin attribute of **IGP**.
- **From Router E**, the route for reaching **150.10.0.0** has an AS_path of 100 and an origin attribute of **IGP**.
- **From Router E**, the route for reaching **190.10.0.0** has an AS_path of 100 and an origin attribute of **Incomplete** (because **190.10.0.0** is a **redistributed** route)

The ORIGIN attribute

- Use a route map and the the **set origin** command to manipulate the ORIGIN attribute.

```
route-map SETORIGIN permit 10
   set origin igp
```

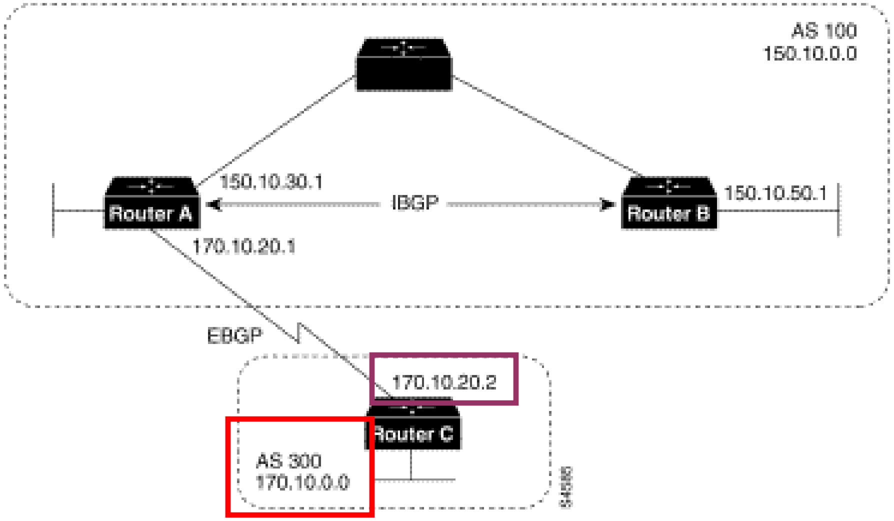
Path Attributes



Attribute Code	Type
1-ORIGIN	Well-known mandatory
2-AS_PATH	Well-known mandatory
3-NEXT_HOP	Well-known mandatory
4-MULTI_EXIT_DISC	Optional non-transitive
5-LOCAL_PREF	Well-known discretionary
6-ATOMIC_AGGREGATE	Well-known discretionary
7-AGGREGATOR	Well-known discretionary
8-COMMUNITY	Optional transitive (Cisco)
9-ORIGINATOR_ID	Optional non-transitive (Cisco)
10-Cluster List	Optional non-transitive (Cisco)
11-Destination Preference	(MCI)
12-Advertiser	(Baynet)
13-rcid_path	(Baynet)
255-Reserved	[md]

NEXT_HOP

- The **NEXT_HOP** attribute is a well-known mandatory attribute (type code 3).
- In terms of an **IGP**, such as RIP, the “next hop” to reach a route is the IP address of the router that has announced the route.
  - Note: The abbreviation **IGP** (Interior Gateway Protocol) will always be in **green**, so not to get it confused with **IBGP** (Interior BGP)
- The **NEXT_HOP** concept with BGP is slightly more elaborate.
- For **EBGP** sessions, the next hop is **the IP address of the neighbor that announced the route**
- For **IBGP** sessions, **for routes originated inside the AS, the next-hop is the IP address of the neighbor that announced the route**.
- **For routes injected into the AS via EBGP, the next hop learned from EBGP is carried unaltered into IBGP**.
  - The next hop is the IP address of the EBGP neighbor from which the route was learned.

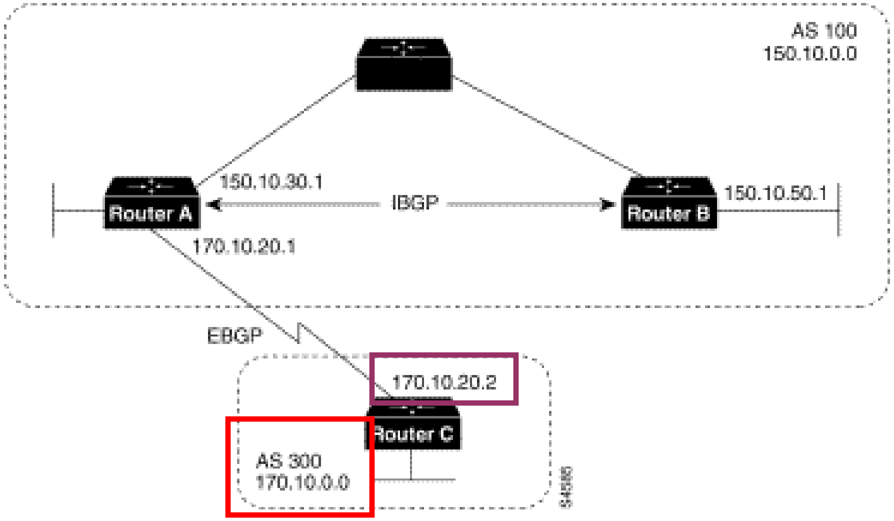


```
Router A
router bgp 100
   neighbor 170.10.20.2 remote-as 300
   neighbor 150.10.50.1 remote-as 100
   network 150.10.0.0

Router B
router bgp 100
   neighbor 150.10.30.1 remote-as 100

Router C
router bgp 300
   neighbor 170.10.20.1 remote-as 100
   network 170.10.0.0
```

- Router C advertises network **170.10.0.0** to Router A with a next hop attribute of **170.10.20.2**, and Router A advertises network 150.10.0.0 to Router C with a next hop attribute of **170.10.20.1**.
- BGP specifies that the ***next hop of EBGP-learned routes should be carried without modification into IBGP***.
- Because of that rule, **Router** A advertises **170.10.0.0** to its IBGP peer (Router B) with a next hop attribute of **170.10.20.2**.
- As a result, according to **Router** B, the next hop to reach **170.10.0.0** is **170.10.20.2**, instead of 150.10.30.1.
- For that reason, the configuration must ensure that **Router B** can reach **170.10.20.2** via an **IGP**.
- Otherwise, Router B will drop packets destined for **170.10.0.0** because the next hop address is inaccessible.
- For example, if Router B runs IGRP, Router A should run IGRP on network **170.10.0.0**.
- You might want to make IGRP passive on the link to Router C so that only BGP updates are exchanged.



Summarize

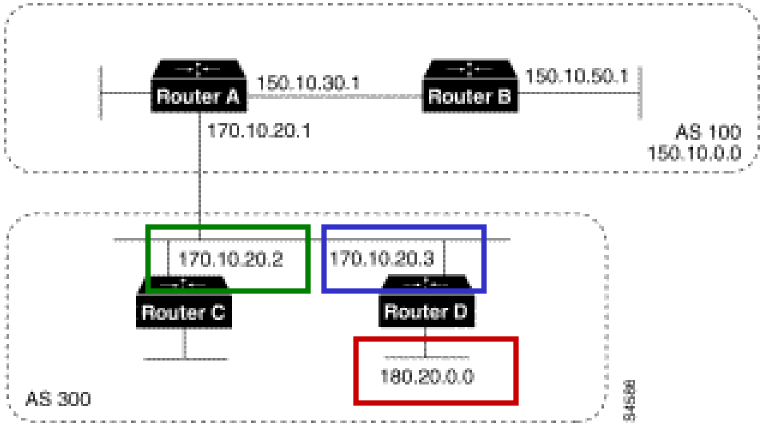
- Router C advertises **170.10.0.0** to Router A with a next hop attribute of **170.10.20.2**,
- Router A advertises **170.10.0.0** to Router B with a next hop attribute of **170.10.20.2**.
- ***The next hop of EBGP-learned routes is passed to the IBGP neighbor without modification into IBGP***.

```
Router A
router bgp 100
   neighbor 170.10.20.2 remote-as 300
   neighbor 150.10.50.1 remote-as 100
   network 150.10.0.0
```

```
Router B
router bgp 100
   neighbor 150.10.30.1 remote-as 100
```

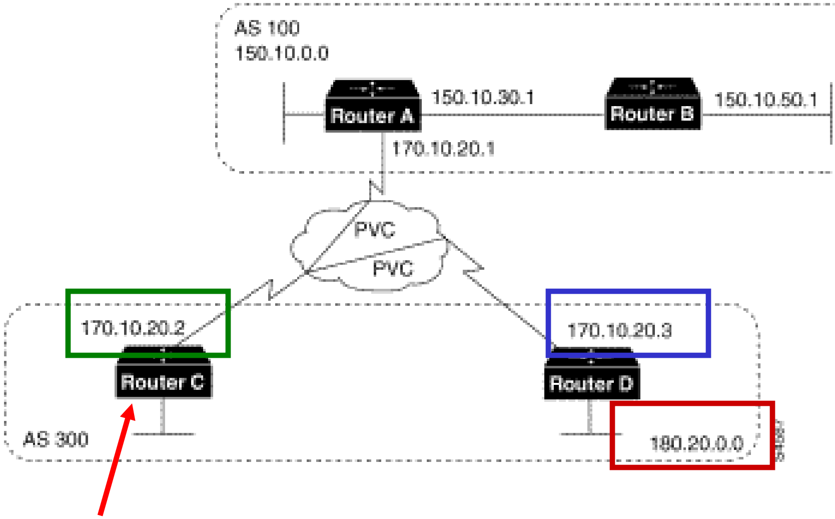
```
Router C
router bgp 300
   neighbor 170.10.20.1 remote-as 100
   network 170.10.0.0
```

NEXT_HOP and Multiaccess Media



- Routers C and D are in AS 300 are running OSPF.
- Router C is running BGP with Router A.
- Router C can reach network **180.20.0.0** via **170.10.20.3**.
- When Router C sends a BGP update to Router A regarding **180.20.0.0**, it sets the next hop attribute to **170.10.20.3**, instead of its own IP address (**170.10.20.2**).
- This is because Routers A, B, and C are in the same subnet, and it makes more sense for Router A to use Router D as the next hop rather than taking an extra hop via Router C.

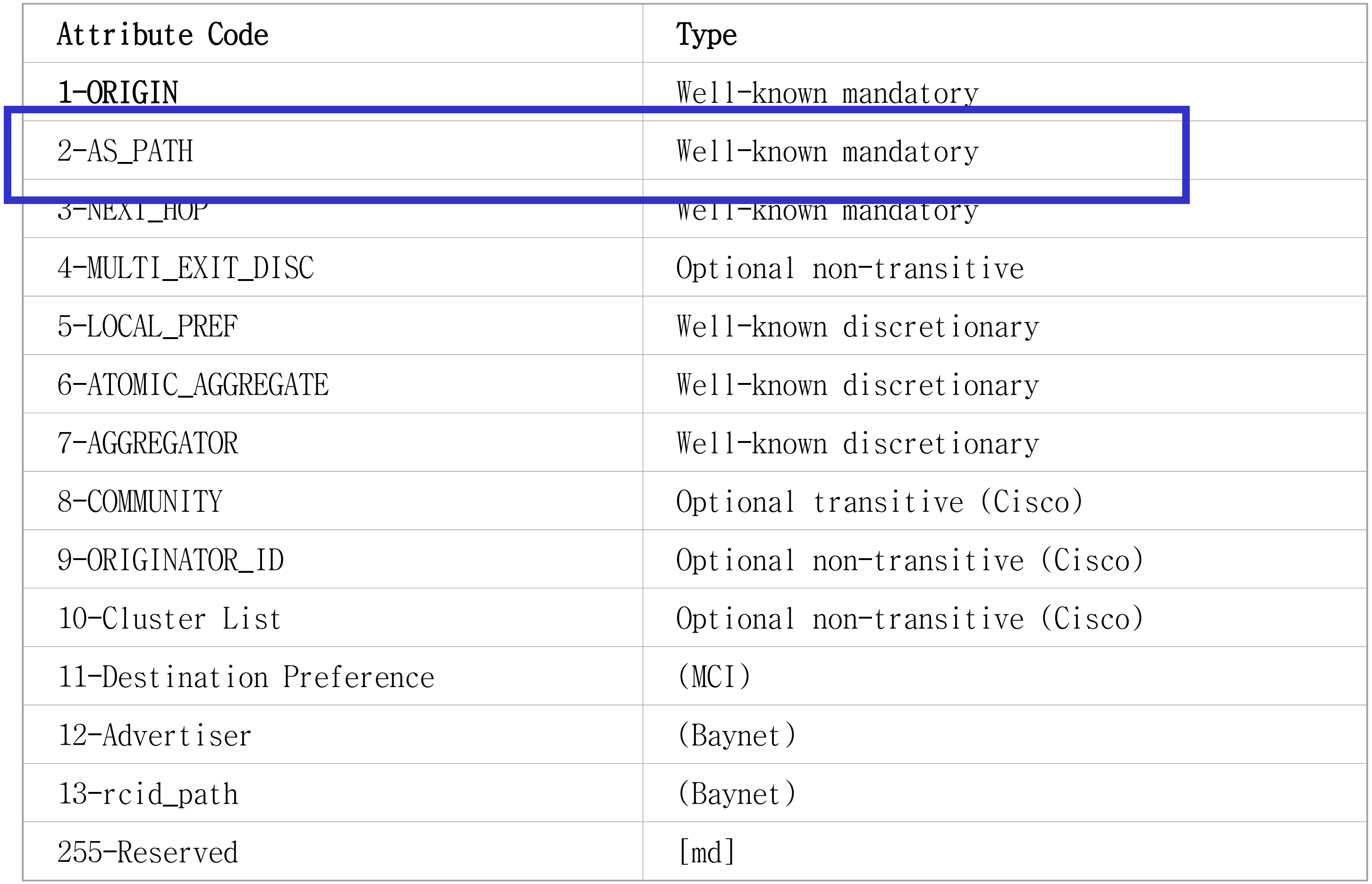
NEXT_HOP and Multiaccess Media



```
Router C
router bgp 300
 neighbor 170.10.20.1 remote-as 100
 neighbor 170.10.20.1 next-hop-self
```

- Routers A, C, and D, use a common media such as Frame Relay (or any NBMA cloud).
- Router C advertises **180.20.0.0** to Router A with a next hop of **170.10.20.3**, just as it would do if the common media were Ethernet.
- The problem is that Router A does not have a direct permanent virtual connection (PVC) to Router D and cannot reach the next hop, so routing will fail.
- To remedy this situation, use the **neighbor next-hop-self** router configuration command**.**
- The neighbor **next-hop-self** command causes Router C to advertise **180.20.0.0** with the next hop attribute set to **170.10.20.2**.

Path Attributes

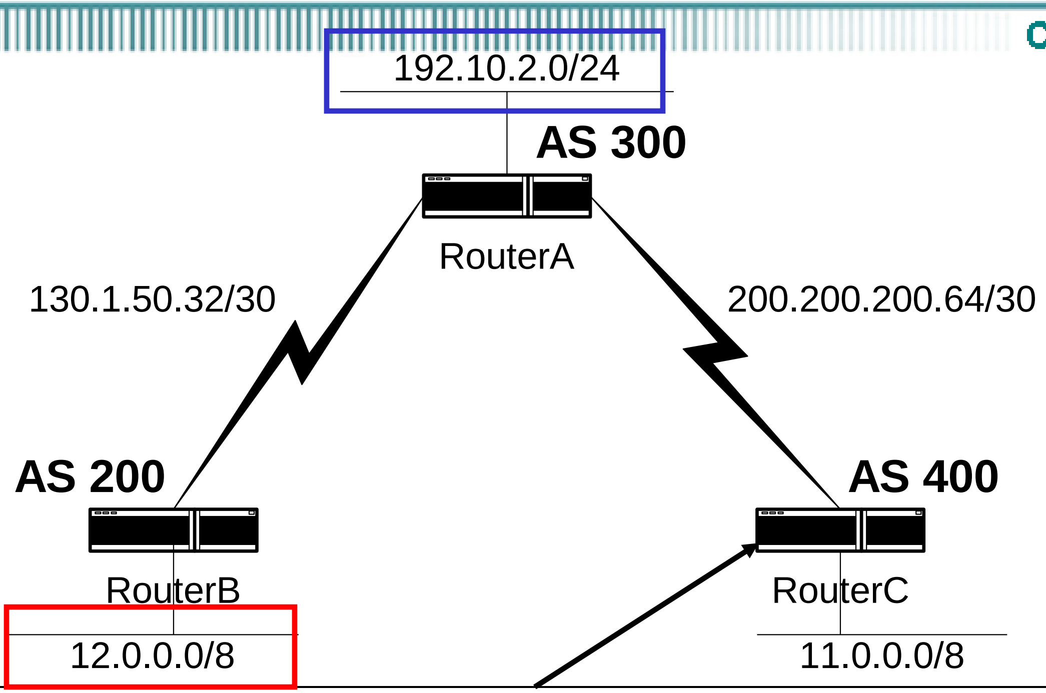


Attribute Code	Type
1-ORIGIN	Well-known mandatory
2-AS_PATH	Well-known mandatory
3-NEXT_HOP	Well-known mandatory
4-MULTI_EXIT_DISC	Optional non-transitive
5-LOCAL_PREF	Well-known discretionary
6-ATOMIC_AGGREGATE	Well-known discretionary
7-AGGREGATOR	Well-known discretionary
8-COMMUNITY	Optional transitive (Cisco)
9-ORIGINATOR_ID	Optional non-transitive (Cisco)
10-Cluster List	Optional non-transitive (Cisco)
11-Destination Preference	(MCI)
12-Advertiser	(Baynet)
13-rcid_path	(Baynet)
255-Reserved	[md]

AS_PATH

- An **AS_PATH** attribute is a well-known mandatory attribute (type code 2).
- It is the **sequence of AS numbers a route has traversed to reach a destination**.
- The AS that originates the route adds its own AS number when sending the route to its external BGP peers.
- Thereafter, each AS that receives the route and passes it on to other BGP peers will prepend its own AS number to the list.
- **Prepending** is the act of adding the AS number to the beginning of the list.
- The **final list** represents all the AS numbers that a route has traversed with the AS number of the AS that originated the route all the way at the end of the list.
- This type of **AS_PATH** list is called an AS_SEQUENCE, because all the AS numbers are ordered sequentially.

AS_PATH



```
RouterC#show ip bgp
BGP table version is 8, local router ID is 200.200.200.66
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         200.200.200.65                         0 300 200 i
*> 192.10.2.0        200.200.200.65          0             0 300 i
```

AS_PATH – private AS numbers

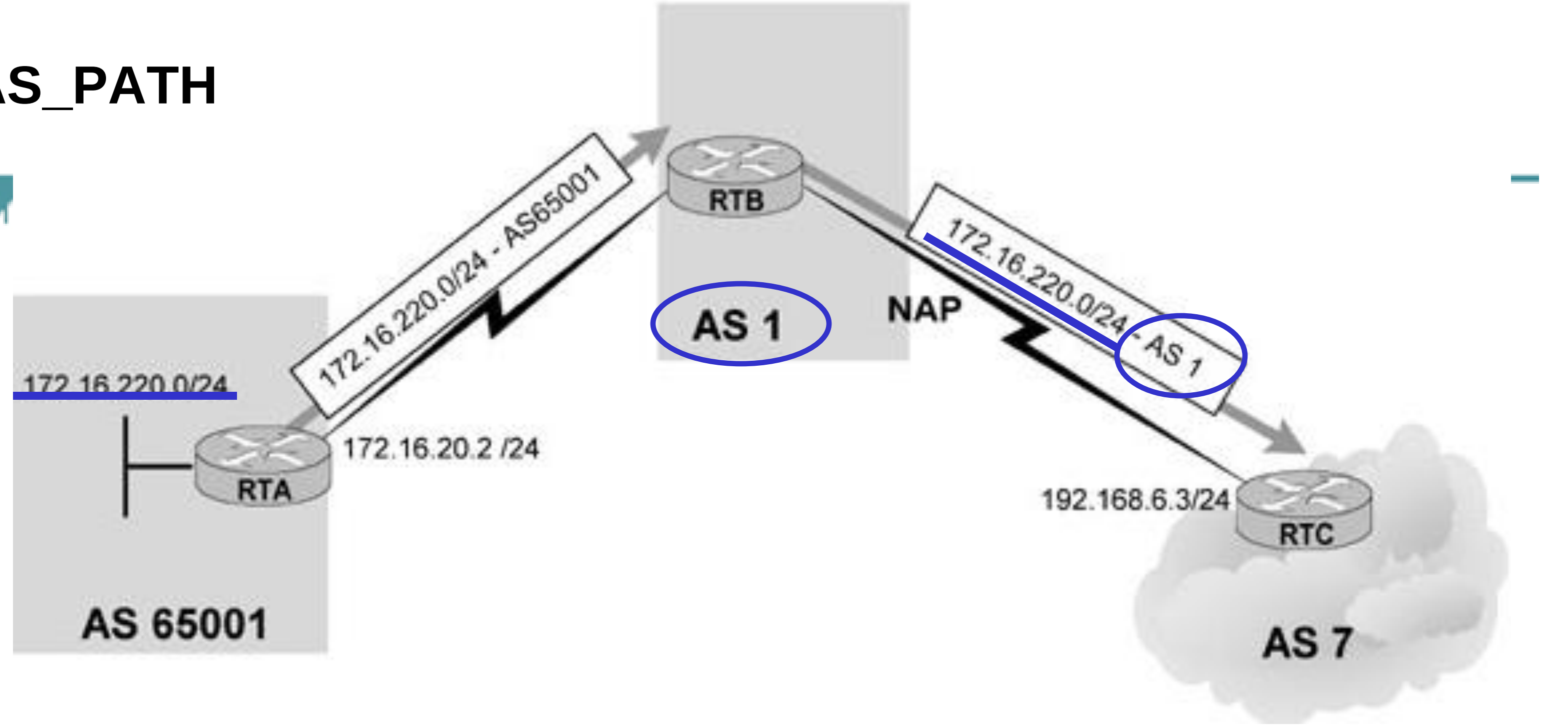
- BGP uses the **AS_PATH** attribute as part of the routing updates (**UPDATE packet**) to ensure a loop-free topology on the Internet.
- Each route that gets passed between BGP peers will carry a list of all AS numbers that the route has already been through.
- If the route is advertised to the AS that originated it, that AS will see itself as part of the **AS_PATH** attribute list and will not accept the route.
- **EBGP**: BGP speakers prepend their AS numbers when advertising routing updates to other autonomous systems (external peers).
- **IBGP**: When the route is passed to a BGP speaker within the same AS, the **AS_PATH** information is left intact.

AS_PATH – private AS numbers

- **AS_PATH** information is one of the attributes BGP looks at to determine the **best route** to take to get to a destination.
  - In comparing two or more different routes, given that all other attributes are identical, a shorter path is always preferred.
  - In case of a tie in AS_PATH length, other attributes are used to make the decision. (later)

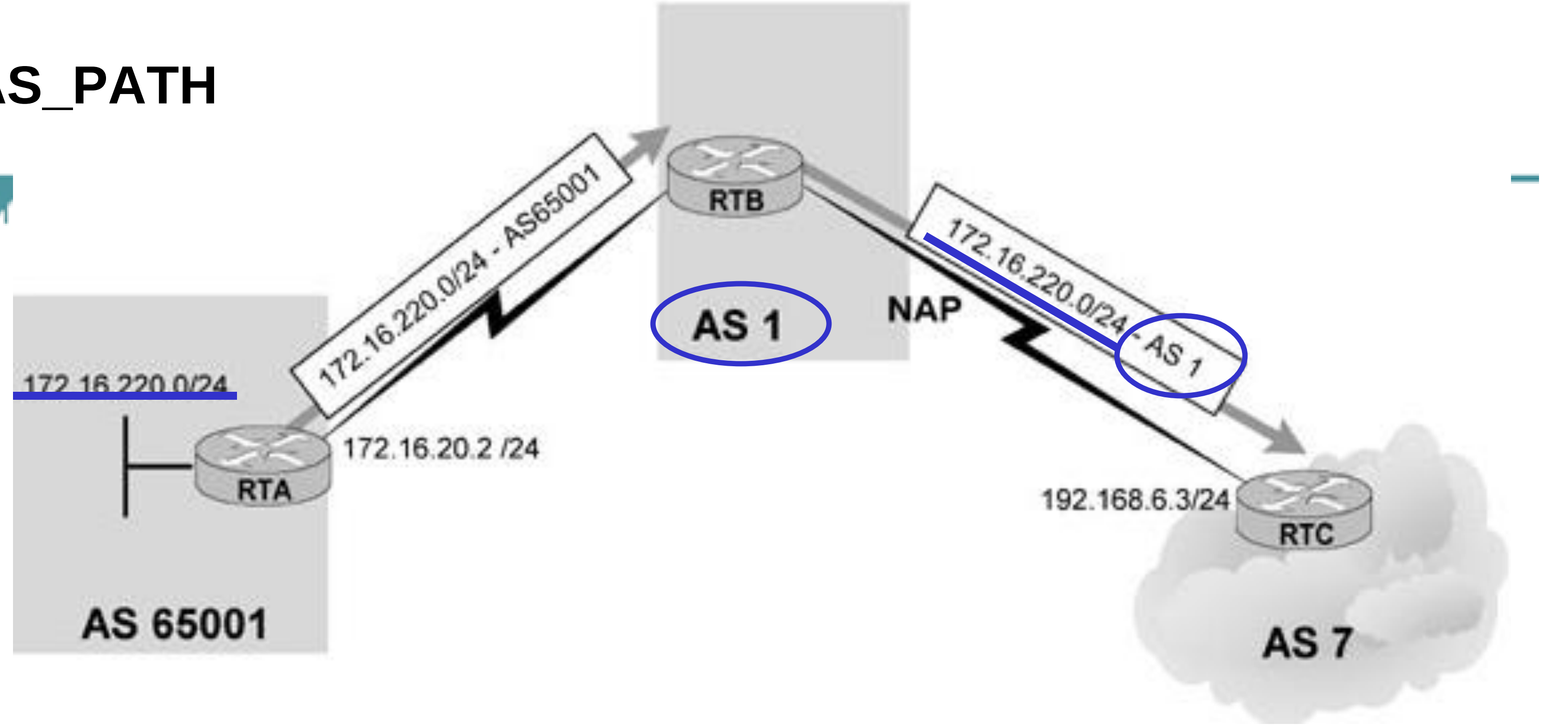
- **Private AS numbers** cannot be leaked to the Internet because they are not unique.
  - Cisco has implemented a feature, **remove-private-as,** to strip private AS numbers out of the **AS_PATH** list before the routes get propagated to the Internet**.**

AS_PATH



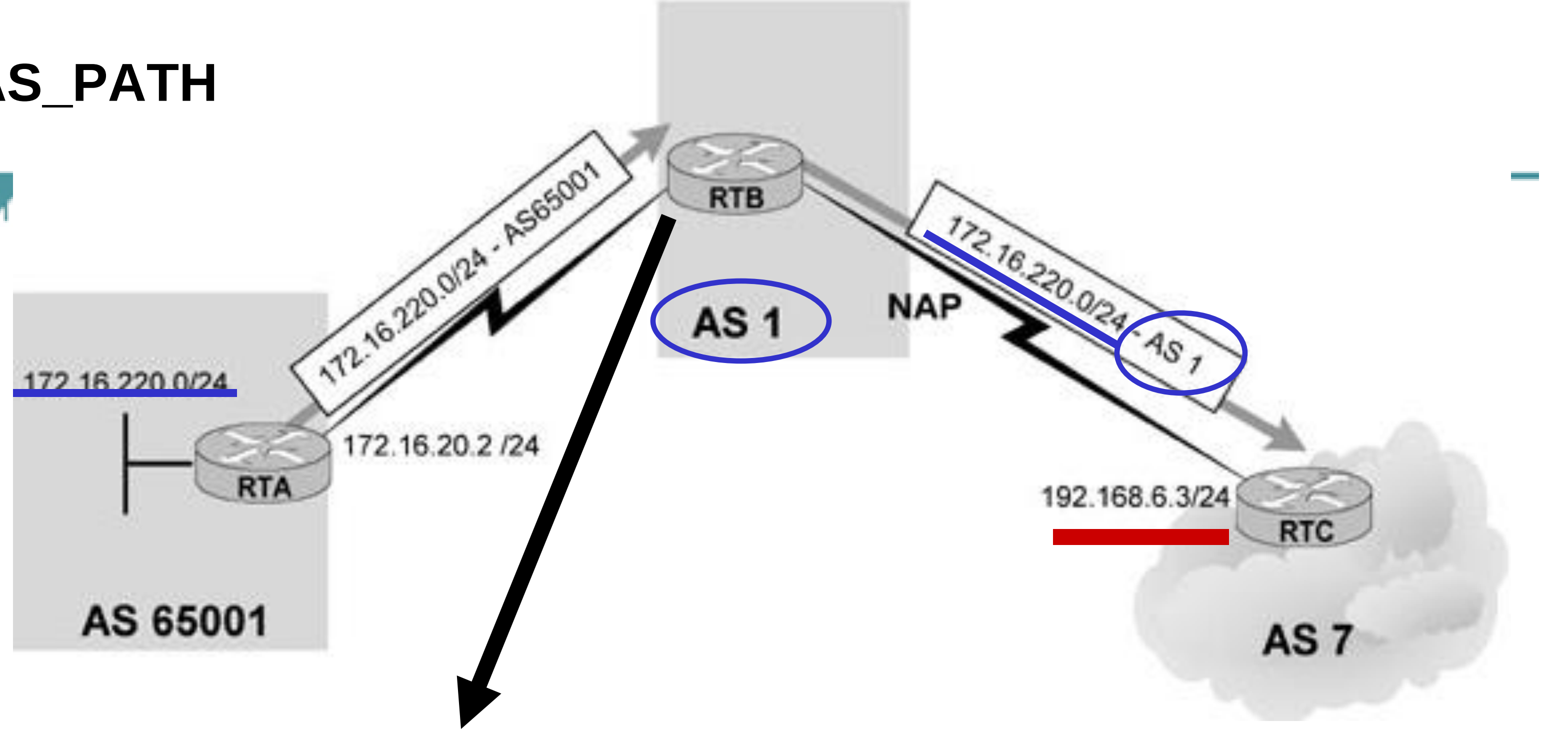
- AS1 is providing Internet connectivity to its customer AS 65001.
- Because the customer connects to only this provider and no plans to connect to an additional provider in the near future, the customer has been allocated a private AS number.
- **BGP will strip private AS numbers** only when propagating updates to the external peers.
- This means that the AS stripping would be configured on RTB as part of its neighbor connection to RTC.

AS_PATH



- Privately numbered autonomous systems should be connected only to a single provider.
- If the **AS_PATH** contains a mixture of private and legal AS numbers, BGP will view this as an illegal design and will **not** strip the private AS numbers from the list, and the update will be treated as usual.
- **"If the AS_PATH includes both private and public AS numbers, BGP doesn't remove the private AS numbers. This situation is considered a configuration error." Cisco**
- Only **AS_PATH** lists that contain private AS numbers in the range 64512 to 65535 are stripped.

AS_PATH



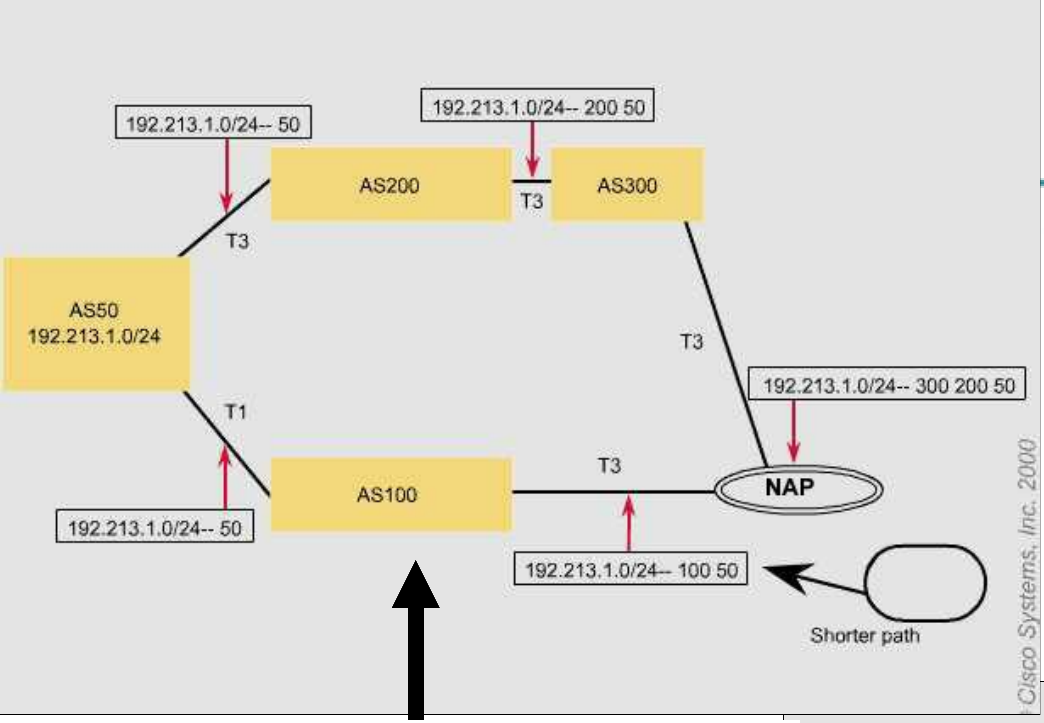
```
RTB(config)#router bgp 1
RTB(config-router)#neighbor 172.16.20.2 remote-as 65001
RTB(config-router)#neighbor 192.168.6.3 remote-as 7
RTB(config-router)#neighbor 192.168.6.3 remove-private-as
```

- Note how RTB is using the **remove-private-as** keyword in its neighbor connection to AS7.
- http://www.cisco.com/warp/public/459/32.html

AS_PATH - prepend

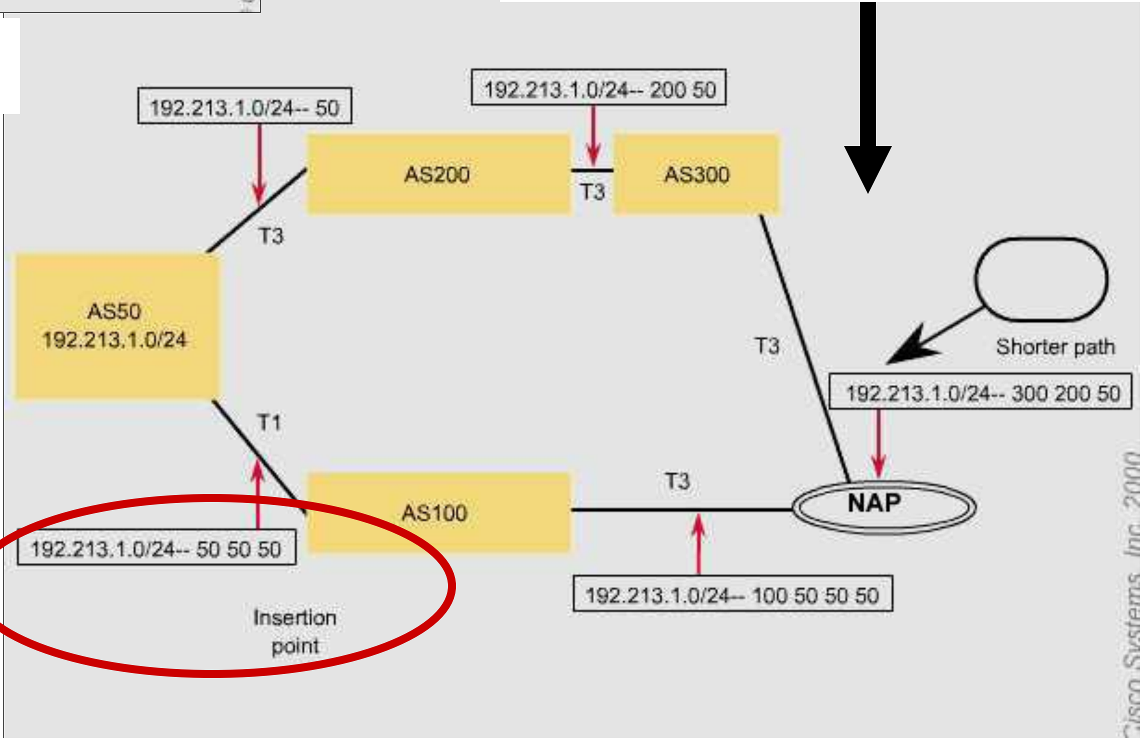
- **AS_PATH** information is manipulated to affect interdomain routing behavior.
- Because BGP prefers a shorter path over a longer one, system operators are tempted to change the path information by including dummy AS path numbers that would increase the path length and influence the traffic trajectory one way or the other.
- **Cisco's implementation** enables a user to **insert AS numbers** at the beginning of an AS_PATH to make the path length longer.

AS_PATH – prepend Concept

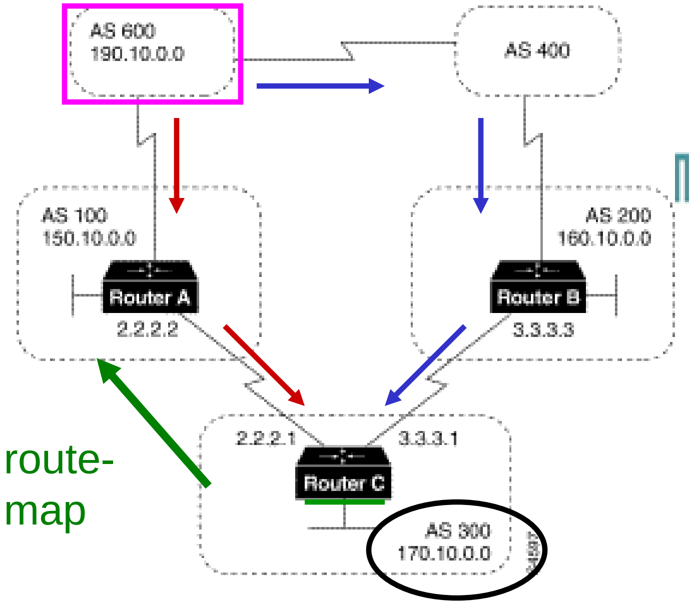


Current “shorter path”

New “shorter path”



AS_PATH – prepend - Example



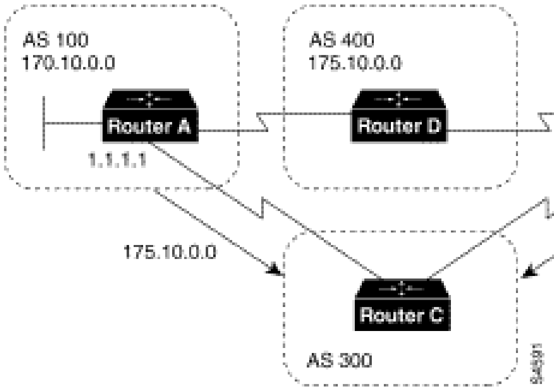
Router C

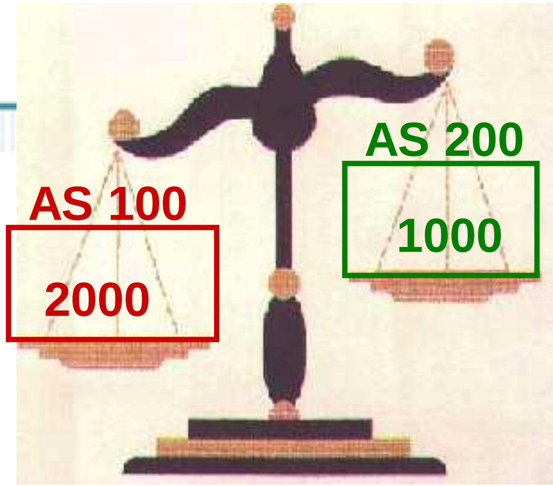
```
router bgp 300
   network 170.10.0.0
   neighbor 3.3.3.3 remote-as 200
   neighbor 2.2.2.2 remote-as 100
   neighbor 2.2.2.2 route-map SETPATH out

route-map SETPATH permit 10
   set as-path prepend 300 300
```

- If you want to **use the configuration of Router C to influence the choice of paths in AS 600**, you can do so by prepending extra AS numbers to the AS_path attribute for routes that **Router C** advertises to AS 100.
- A common practice is to repeat the AS number, as in the above configuration.
- The set as-path route map configuration command with the prepend keyword causes **Router C to prepend 300 twice to the value of the AS_path attribute before it sends updates to the neighbor at IP address 2.2.2.2 (Router A).**
- As a result, the AS_path attribute of updates for network **170.10.0.0** that **AS 600** receives via **AS 100** will be **100, 300, 300, 300**, which is longer than the value of the AS_path attribute of updates for network **170.10.0.0** that **AS 600** receives via **AS 400 (400, 200, 300).**
- **AS 600 will choose (400, 200, 300) as the better path.**

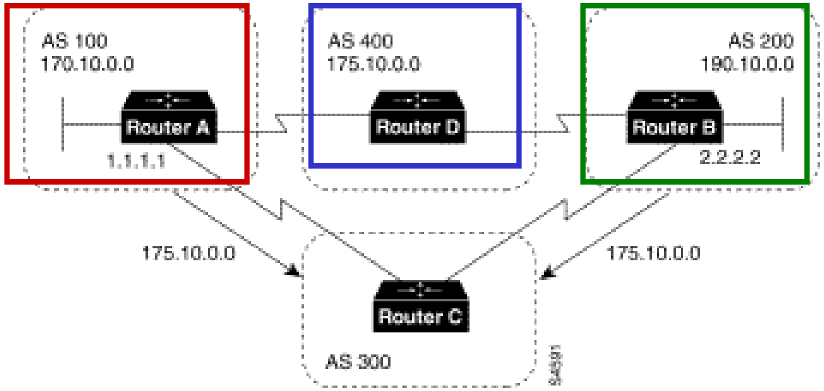
The WEIGHT attribute

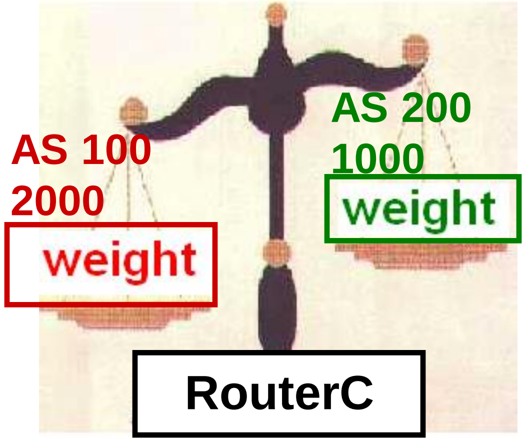




- The weight attribute is a **special Cisco attribute** that is used in the path selection process **when there is more than one route to the same destination**.
- The weight attribute is **local to the router on which it is assigned**, and it is **not propagated** in routing updates.
- By **default**, the weight attribute is **32768** for paths that the router originates and **zero** for other paths.
- Routes with a **higher weight are preferred** when there are multiple routes to the same destination.

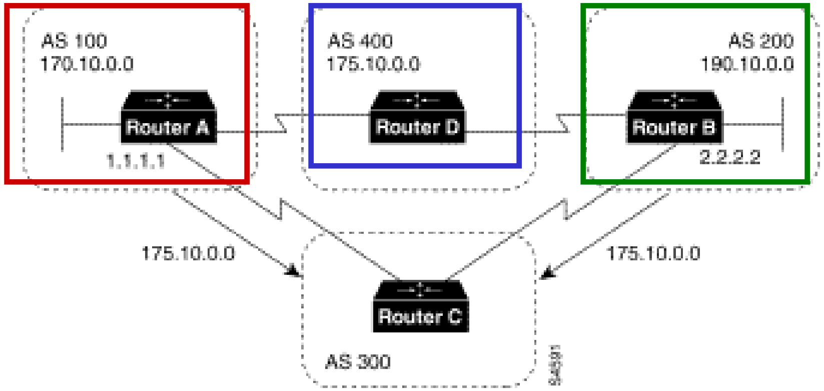
The WEIGHT attribute

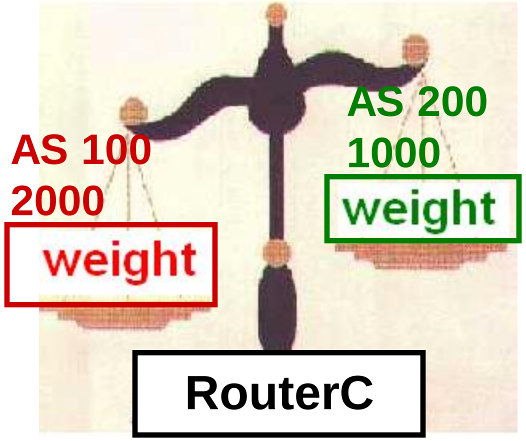




- **Router A** and **Router B** learn about network **175.10.0.0 from AS 400**, and each propagates the update to **Router C**.
- **Router C** has two routes for reaching **175.10.0.0** and has to decide which route to use.
- If, on **Router C**, you set the weight of the updates coming in from **Router A** to be higher than the updates coming in from **Router B**, **Router C** will use **Router A** as the next hop to reach network **175.10.0.0**.

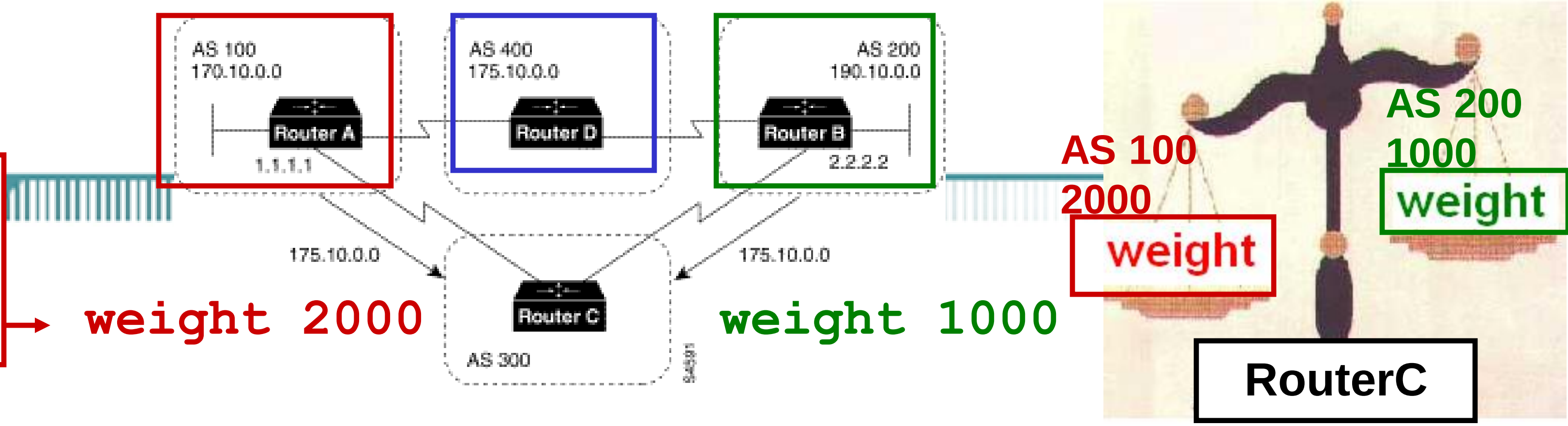
The WEIGHT attribute





- There are three ways to set the weight for updates coming in from Router A:
  - Using the **neighbor weight** Command to Set the Weight Attribute
    - What we will use.
    - Because of time reasons, we will only discuss this option.
  - Using an **Access List** to Set the Weight Attribute
    - FYI
  - Using a **Route Map** to Set the Weight Attribute
    - FYI

Higher weight preferred →



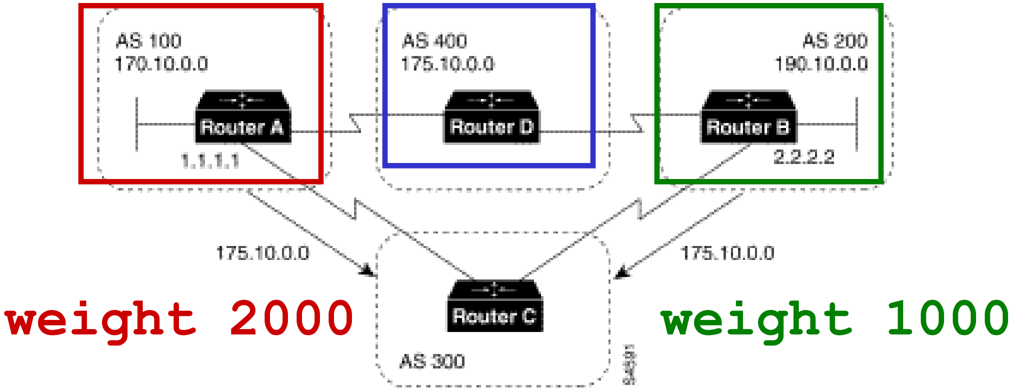
Using the neighbor weight Command to Set the Weight Attribute

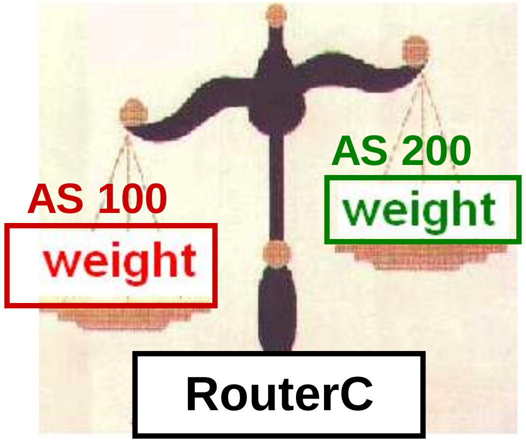
- The following configuration for Router C uses the **neighbor weight** router configuration command:

```
Router C
router bgp 300
   neighbor 1.1.1.1 remote-as 100
   neighbor 1.1.1.1 weight 2000
   neighbor 2.2.2.2 remote-as 200
   neighbor 2.2.2.2 weight 1000
```

- This configuration sets the **weight** of **all route updates from AS 100 to 2000**, and the **weight** of **all route updates coming from AS 200 to 1000**.
- **Result**: The higher **weight** assigned to route updates from AS 100 causes **Router C** to send traffic through **Router A**.

The WEIGHT attribute





Using an Access List to Set the Weight Attribute - FYI

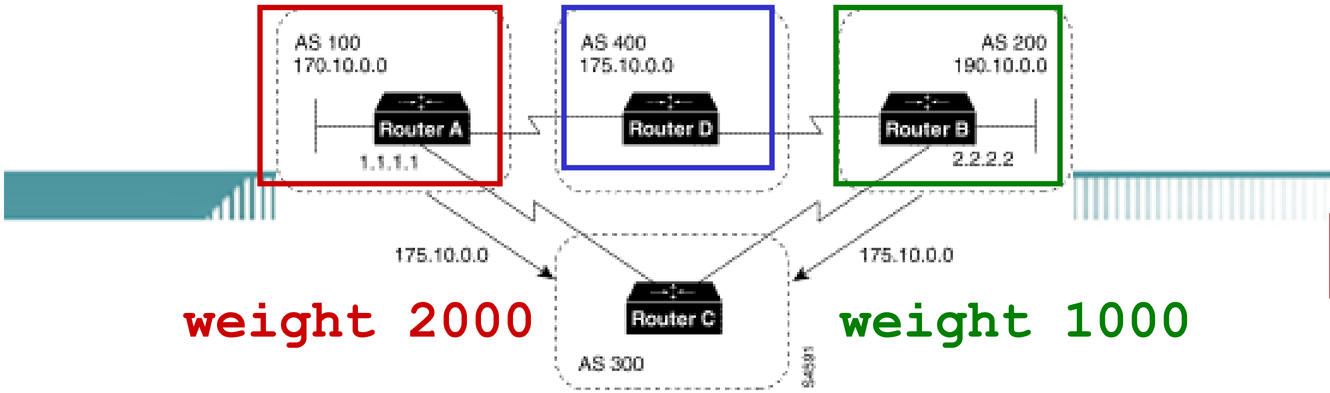
- The following commands on Router C use access lists and the value of the AS_path attribute to assign a weight to route updates:

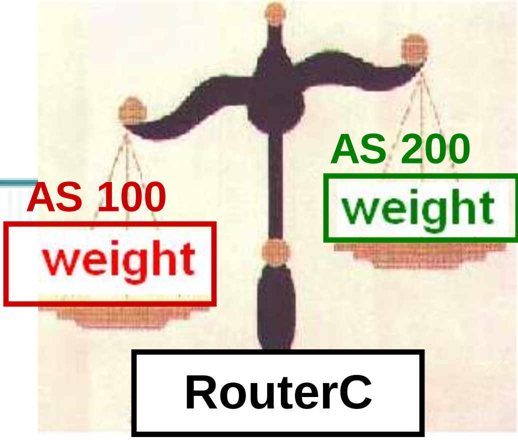
```
Router C
router bgp 300
   neighbor 1.1.1.1 remote-as 100
   neighbor 1.1.1.1 filter-list 5 weight 2000
   neighbor 2.2.2.2 remote-as 200
   neighbor 2.2.2.2 filter-list 6 weight 1000
ip as-path access-list 5 permit ^100$
ip as-path access-list 6 permit ^200$
```

The WEIGHT attribute – Access list FYI

```
Router C
router bgp 300
   neighbor 1.1.1.1 remote-as 100
   neighbor 1.1.1.1 filter-list 5 weight 2000
   neighbor 2.2.2.2 remote-as 200
   neighbor 2.2.2.2 filter-list 6 weight 1000
ip as-path access-list 5 permit ^100$
ip as-path access-list 6 permit ^200$
```

- In this example, **2000 is assigned to the weight attribute of updates from the neighbor at IP address 1.1.1.1 that are permitted by access list 5.**
- Access list 5 permits updates whose AS_path attribute starts with 100 (as specified by ^) and ends with 100 (as specified by $). (The ^ and $ symbols are used to form regular expressions. For a complete explanation of regular expressions, see the appendix on regular expressions in the Cisco Internetwork Operating System (Cisco IOS) software configuration guides and command references.
- This example also **assigns 1000 to the weight attribute of updates from the neighbor at IP address 2.2.2.2 that are permitted by access list 6**. Access list 6 permits updates whose AS_path attribute starts with 200 and ends with 200.
- In effect, this configuration assigns 2000 to the weight attribute of all route updates received from AS 100 and assigns 1000 to the weight attribute of all route updates from AS 200.

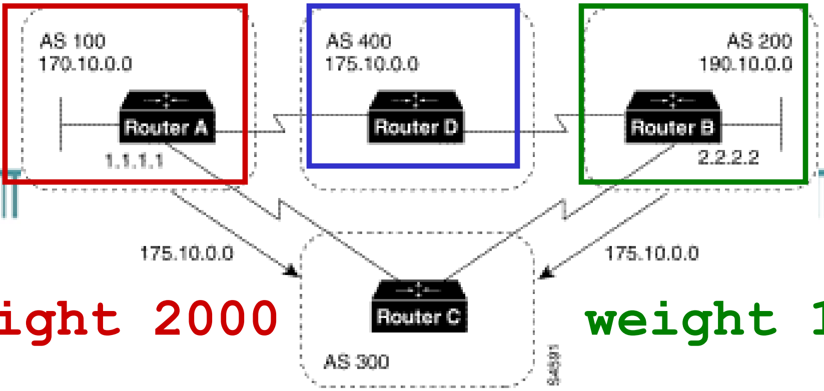




Using a Route Map to Set the Weight Attribute - FYI

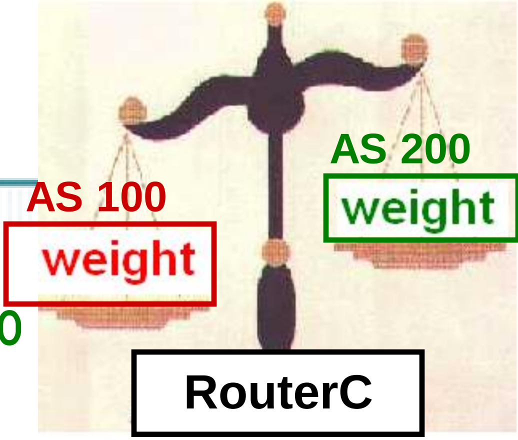
- The following commands on Router C use a route map to assign a weight to route updates:

```
Router C
router bgp 300
   neighbor 1.1.1.1 remote-as 100
   neighbor 1.1.1.1 route-map SETWEIGHTIN in
   neighbor 2.2.2.2 remote-as 200
   neighbor 2.2.2.2 route-map SETWEIGHTIN in
ip as-path access-list 5 permit ^100$
route-map SETWEIGHTIN permit 10
   match as-path 5
   set weight 2000
route-map SETWEIGHTIN permit 20
   set weight 1000
```



weight 2000

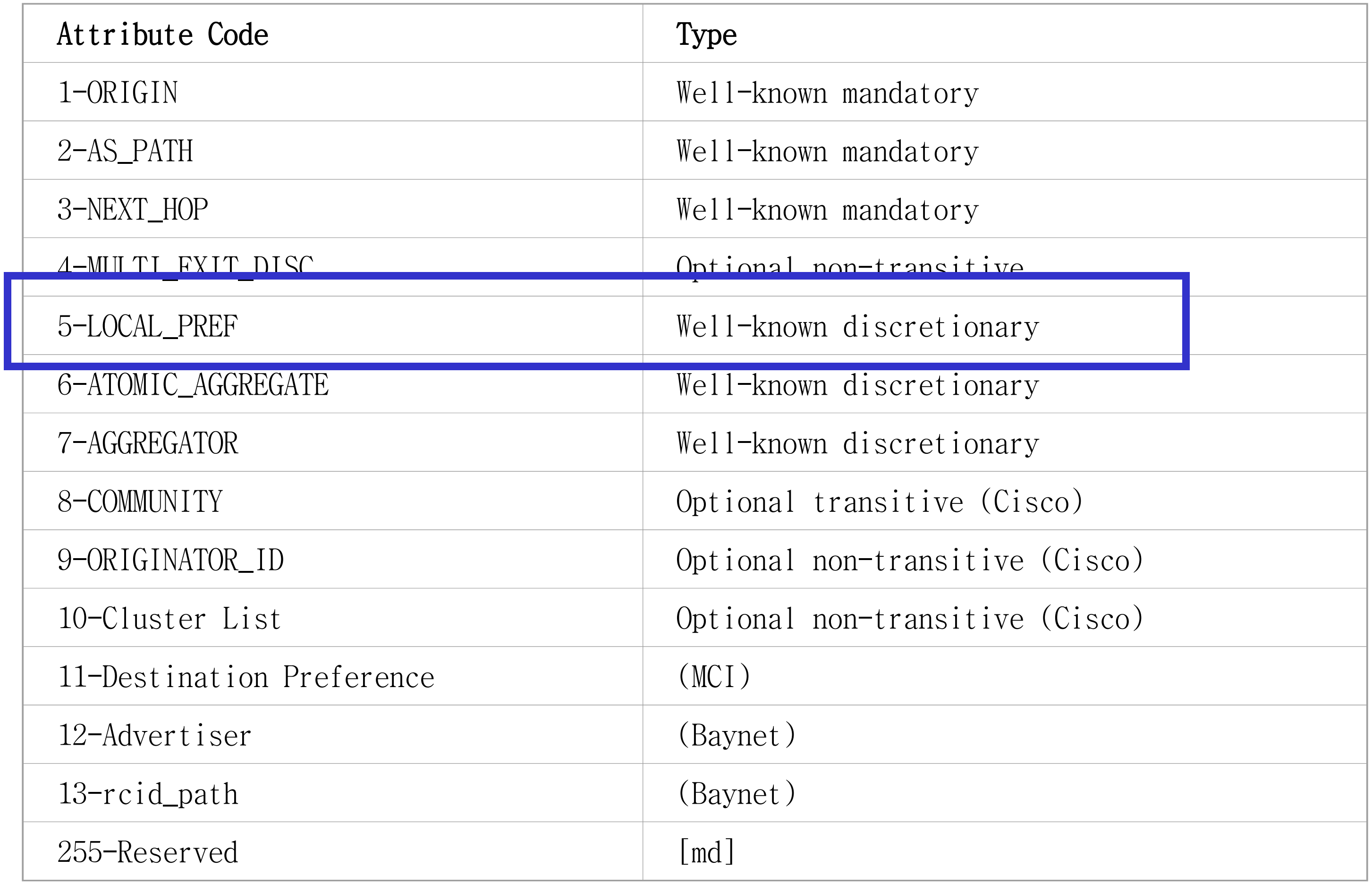
weight 1000



```
Router C
router bgp 300
   neighbor 1.1.1.1 remote-as 100
   neighbor 1.1.1.1 route-map SETWEIGHTIN in
   neighbor 2.2.2.2 remote-as 200
   neighbor 2.2.2.2 route-map SETWEIGHTIN in
ip as-path access-list 5 permit ^100$
route-map SETWEIGHTIN permit 10
   match as-path 5
   set weight 2000
route-map SETWEIGHTIN permit 20
   set weight 1000
```

- This first instance of the setweightin route map assigns 2000 to any route update from AS 100, and the second instance of the setweightin route map assigns 1000 to route updates from any other AS

Path Attributes

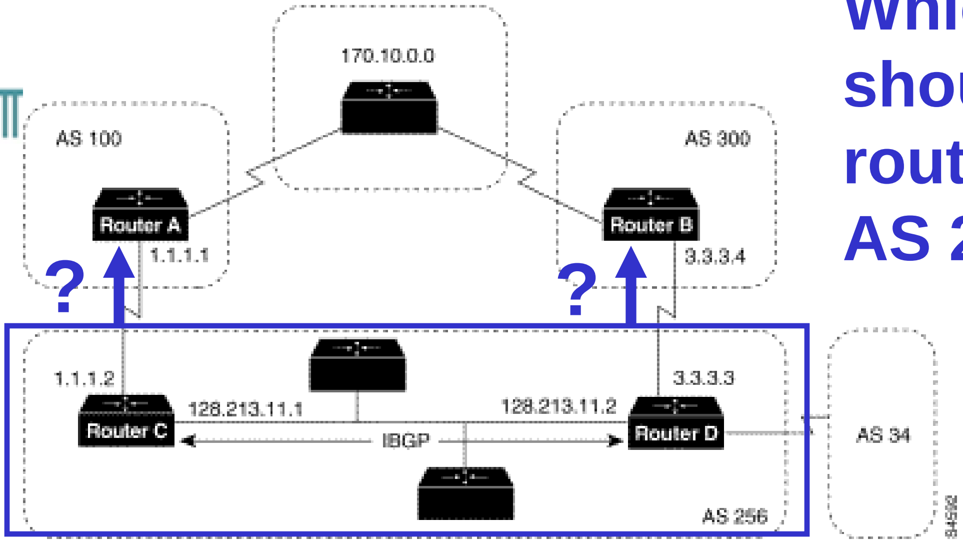


Attribute Code	Type
1-ORIGIN	Well-known mandatory
2-AS_PATH	Well-known mandatory
3-NEXT_HOP	Well-known mandatory
4-MULTI_EXIT_DISC	Optional non-transitive
5-LOCAL_PREF	Well-known discretionary
6-ATOMIC_AGGREGATE	Well-known discretionary
7-AGGREGATOR	Well-known discretionary
8-COMMUNITY	Optional transitive (Cisco)
9-ORIGINATOR_ID	Optional non-transitive (Cisco)
10-Cluster List	Optional non-transitive (Cisco)
11-Destination Preference	(MCI)
12-Advertiser	(Baynet)
13-rcid_path	(Baynet)
255-Reserved	[md]

The LOCAL_PREF Attribute

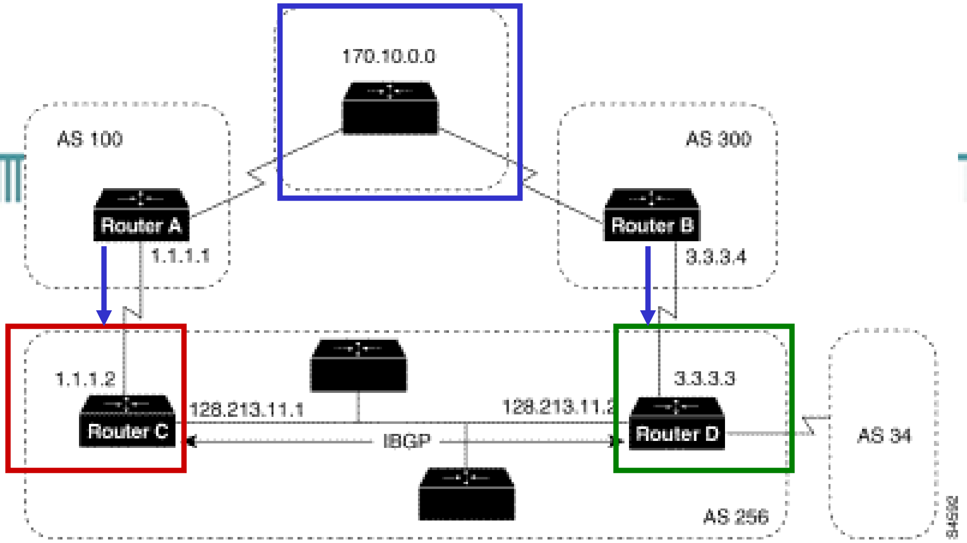
- Well-known discretionary attribute (type code 5).
- Degree of preference given to a route to compare it with other routes for the same destination
  - Higher LOCAL_PREF values are preferred
- **Local to the AS**
  - Exchanged between IBGP peers only
  - It is not advertised to EBGP peers
- **Routers within a multi-homed AS may learn that they can reach the same destination network via neighbors in two (or more) different autonomous systems**.
  - There could be two or more exit points from the local AS to any given destination.
- You can use the **LOCAL_PREF** attribute **to force your BGP routers to prefer one exit point over another** when routing to a particular destination network.

The LOCAL_PREF Attribute

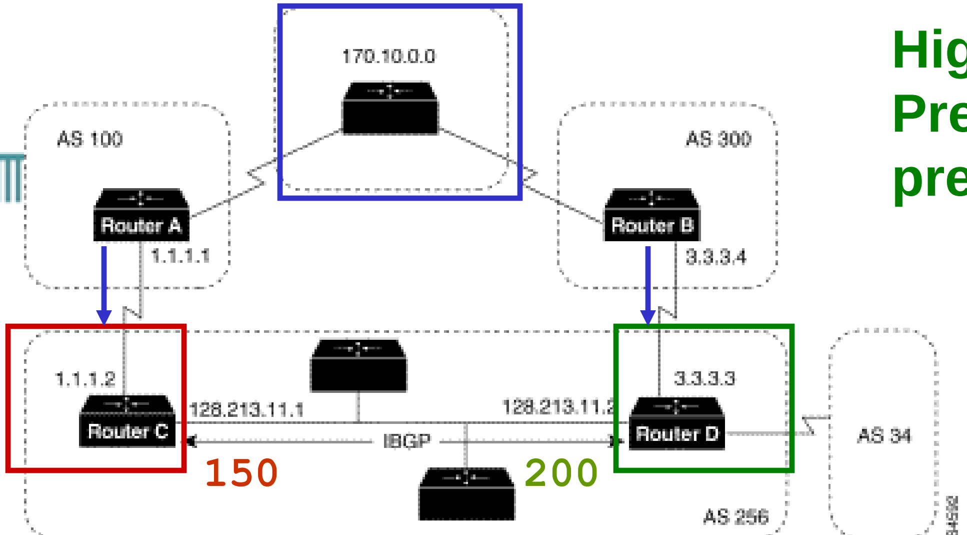


Which exit should all the routers within AS 256 use?

- Because this attribute is communicated within all BGP routers inside the AS, **all BGP routers will have a common view on how to exit the AS**.
- Although routers always prefer the **lowest-route metric and administrative distance** for a given destination, **BGP routers prefer higher LOCAL_PREF values over lower ones**.
- When there are **multiple paths to the same destination, the local preference attribute indicates the preferred path**.
- The path with the **higher preference is preferred** (the **default** value of the **local preference** attribute is **100**).
- Unlike the **weight attribute**, which is only **relevant to the local router**, the **local preference attribute** is part of the routing update and is **exchanged among routers in the same AS.**



- AS 256 receives route updates for network **170.10.0.0** from **AS 100** and **AS 300**.
- There are two ways to set local preference:
  - Using the **bgp default local-preference** command
  - Using a **Route Map** to Set Local Preference - **FYI**



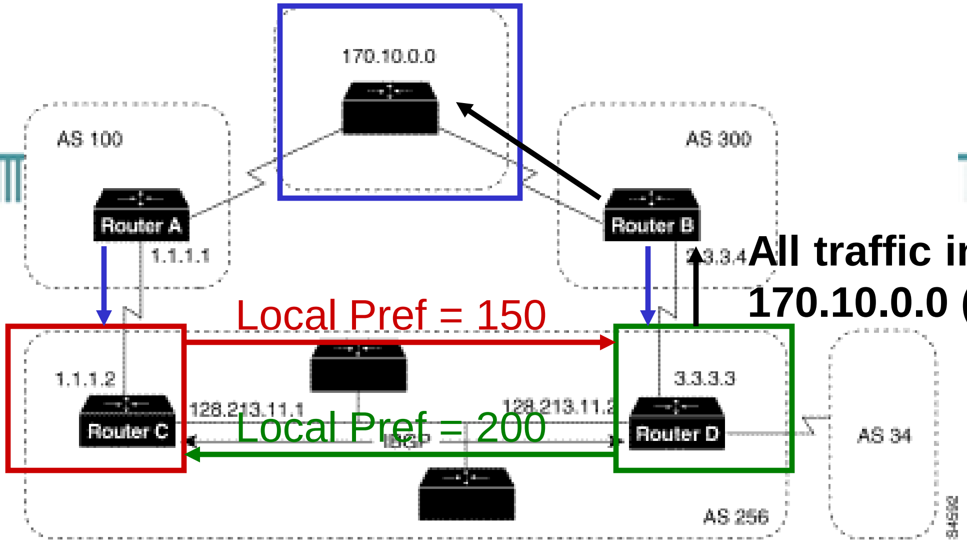
Higher Local Preference is preferred!

Using the bgp default local-preference Command

- The following configurations use the bgp default local-preference router configuration command to set the local preference attribute on Routers C and D:

```
Router C
router bgp 256
   neighbor 1.1.1.1 remote-as 100
   neighbor 128.213.11.2 remote-as 256
   bgp default local-preference 150
Router D
router bgp 256
   neighbor 3.3.3.4 remote-as 300
   neighbor 128.213.11.1 remote-as 256
   bgp default local-preference 200
```

Higher Local Preference is preferred!

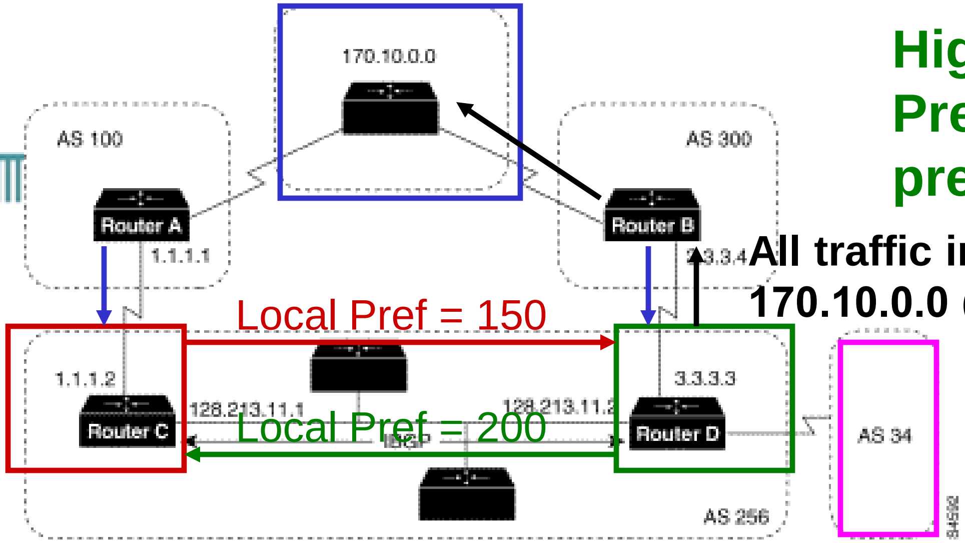


All traffic in AS 256 destined for 170.10.0.0 (and other ASes)

```
Router C
router bgp 256
   bgp default local-preference 150
Router D
router bgp 256
   bgp default local-preference 200
```

Higher Local Preference is preferred!

- The configuration for **Router C** causes it to set the **local preference** of all updates **from AS 300 to 150 (routes learned from RouterD)**, and the configuration for **Router D** causes it to set the **local preference** for all updates **from AS 100 to 200 (routes learned from RouterC)**.
- Because **local preference** is exchanged within the AS, both Routers C and D determine that updates regarding network **170.10.0.0** have a higher **local preference** when they come from **AS 300** than when they come from AS 100.
- As a **result**, **all traffic in AS 256 destined for network 170.10.0.0 is sent to Router D as the exit point**.



Local Pref = 150

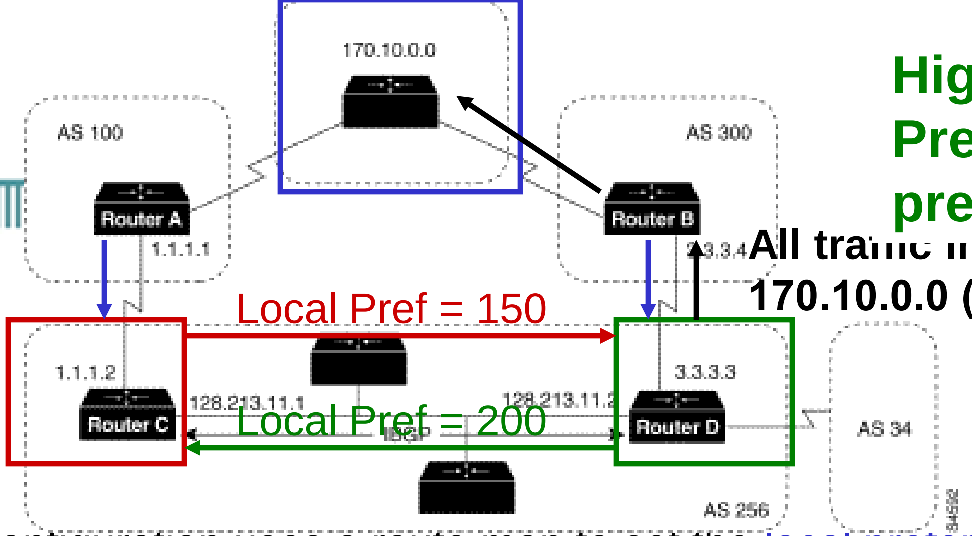
Local Pref = 200

Higher Local Preference is preferred!

All traffic in AS 256 destined for 170.10.0.0 (and other AS's)

Using a Route Map to Set Local Preference - FYI

- **Route maps** provide more flexibility than the bgp default local-preference router configuration command.
- When the bgp default local-preference command is used on Router D, the local preference attribute of **all** updates received by Router D will be set to 200, including updates from **AS 34**.



Local Pref = 150

Local Pref = 200

Higher Local Preference is preferred!

All traffic in AS 256 destined for 170.10.0.0 (and other AS's)

- The following configuration uses a route map to set the local preference attribute on Router D specifically for updates regarding AS 300:

```
Router D
router bgp 256
   neighbor 3.3.3.4 remote-as 300
   route-map SETLOCALIN in
   neighbor 128.213.11.1 remote-as 256
ip as-path 7 permit ^300$
route-map SETLOCALIN permit 10
   match as-path 7
   set local-preference 200
route-map SETLOCALIN permit 20
```

- With this configuration, the local preference attribute of any update coming **from AS 300 is set to 200**.
- Instance 20 of the SETLOCALIN route map accepts all other routes.

AS_PATH Attribute - FYI

```
Router D
router bgp 256
   neighbor 3.3.3.4 remote-as 300
   route-map SETLOCALIN in
   neighbor 128.213.11.1 remote-as 256
ip as-path access-list 7 permit ^300$
route-map SETLOCALIN permit 10
   match as-path 7
   set local-preference 200
route-map SETLOCALIN permit 20
```

This command creates a special kind of access list that looks at the AS_PATH attribute.

- The **ip as-path access-list** command is used with route maps to match part (or all) of a route’s AS PATH.
  - – Regular expressions are used with this command to provide specificity.

AS_PATH Attribute - FYI

```
Router D
router bgp 256
   neighbor 3.3.3.4 remote-as 300
   route-map SETLOCALIN in
   neighbor 128.213.11.1 remote-as 256
ip as-path access-list 7 permit ^300$
route-map SETLOCALIN permit 10
   match as-path 7
   set local-preference 200
route-map SETLOCALIN permit 20
```

- Note that the previous example uses the **ip as-path access-list** command, which here matches the regular expression **^300$**.
- Essentially, this statement **matches any routes that include AS 300 in their AS_PATH attribute**.
- With the configuration, the **LOCAL_PREF** attribute of any update coming from AS 300 is set to 200 by instance 10 of the route map, SETLOCALIN.
- Instance 20 of the route map accepts all other routes.

Regular Expressions - FYI

- A **regular expression** is a pattern to match against an input string.
- The input string, in the case of the **ip as-path access-list** command, is the **AS_PATH attribute**.
- Once you specify a pattern (or patterns) using this command, the router tests BGP routes to see if the AS_PATH attribute matches the pattern or not.
- For example, the following command will match any AS_PATH that includes 2150:

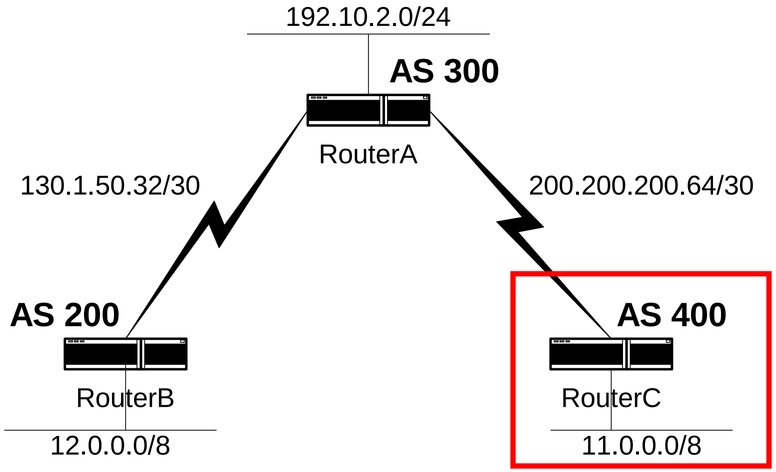
```
Router(config)#ip as-path access-list 1 permit 2150
Or
Router# show ip bgp regexp 2150
```

- Unfortunately, the regular expression, 2150, will match not only AS 2150, but also 12150, 21502, 21503, etc.
- Because policy routing demands a certain degree of precision, you will typically use one or more these special characters when creating a regular expression.

Regular Expressions

- A **regular expression** is a pattern to match against an input string.

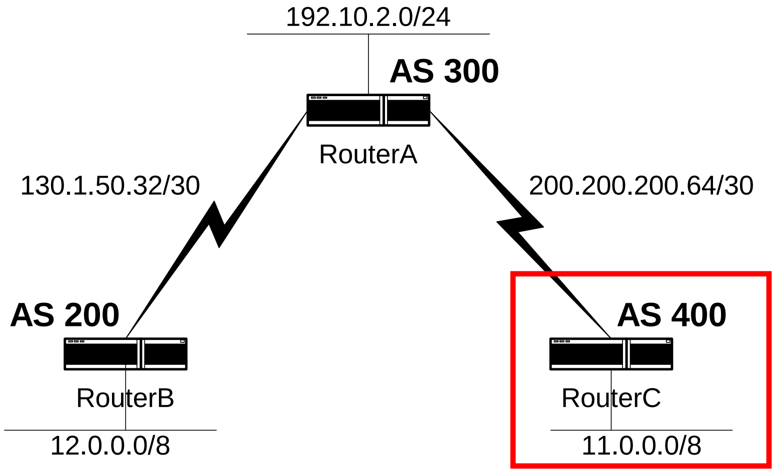
Character	Description
^	Matches the beginning of the input string.
$	Matches the end of the input string.
_	Matches a space, comma, left brace, right brace, the beginning of an input string, or the ending of an input stream
.	Matches any single character
*	Matches 0 or more single- or multiple-character patterns.



```
RouterC#show ip bgp
   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         200.200.200.65                         0 300 200 i
*> 192.10.2.0       200.200.200.65           0             0 300 i
```

```
RouterC# show ip bgp regexp ^300
```

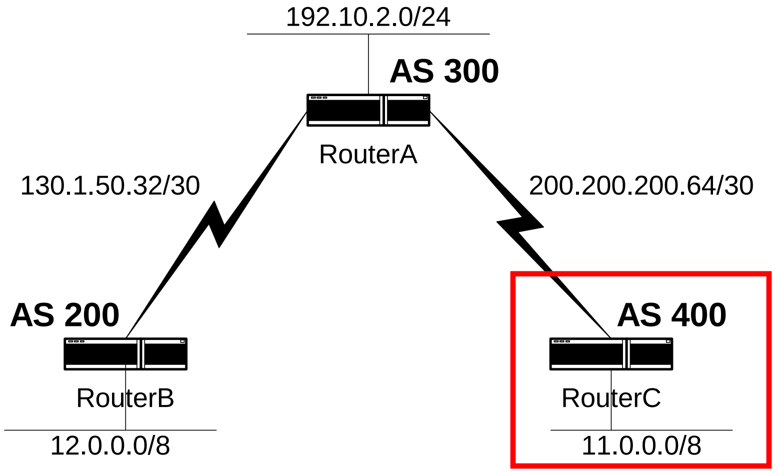
- **Match beginning of input string, AS_PATH, = 300**
- **Last prepended AS was 300:**
- **Routes matched: 12.0.0.0 and 192.10.2.0**



```
RouterC#show ip bgp
   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         200.200.200.65                         0 300 200 i
*> 192.10.2.0       200.200.200.65           0             0 300 i
```

```
RouterC# show ip bgp regexp ^200
```

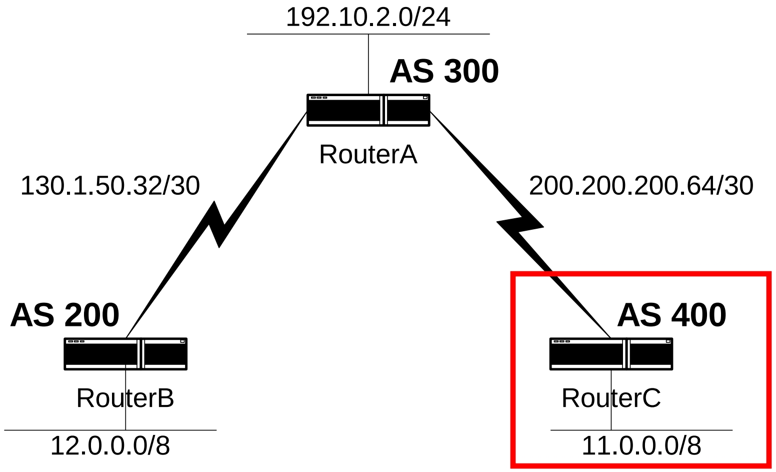
- **Match beginning of input string, AS_PATH, = 200**
- **Last prepended AS was 200:**
- **Routes matched : none**



```
RouterC#show ip bgp
   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         200.200.200.65                         0 300 200 i
*> 192.10.2.0       200.200.200.65           0             0 300 i
```

```
RouterC# show ip bgp regexp 300$
```

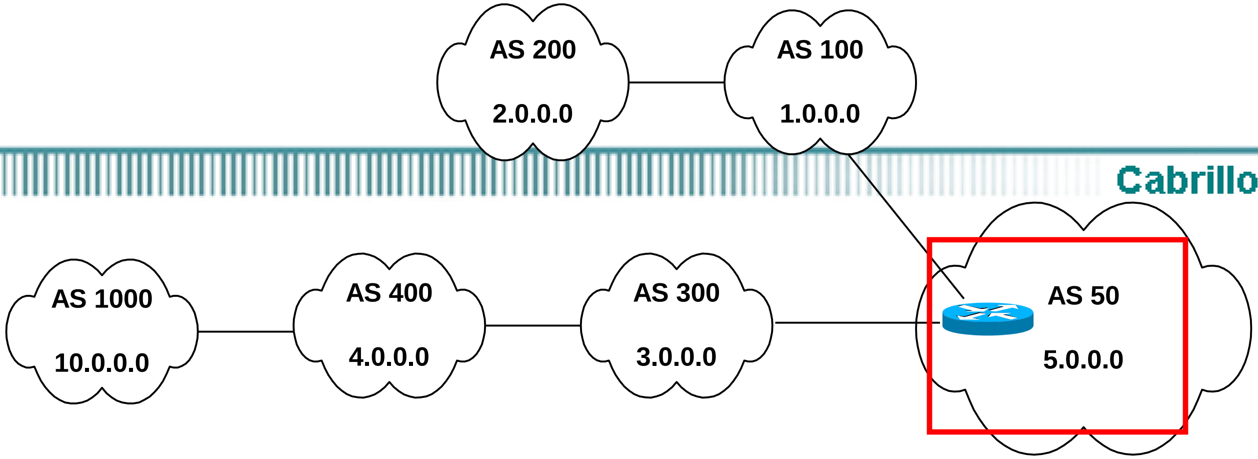
- **Match end of input string, AS_PATH, = 300**
- **Originating AS = 300:**
- **Routes matched : 192.10.2.0**



```
RouterC#show ip bgp
   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         200.200.200.65                         0 300 200 i
*> 192.10.2.0       200.200.200.65           0             0 300 i
```

```
RouterC# show ip bgp regexp 200$
```

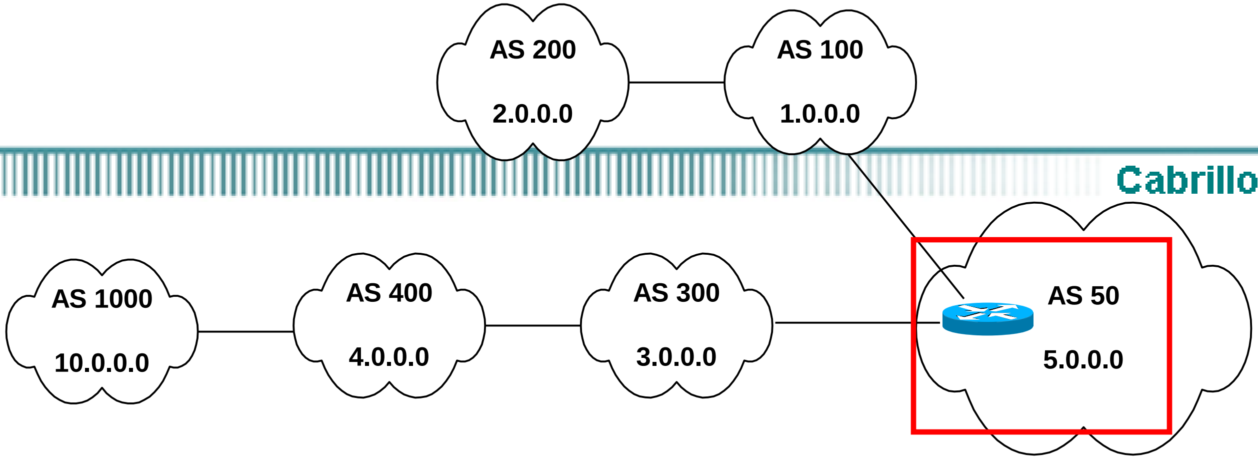
- **Match end of input string, AS_PATH, = 200**
- **Originating AS = 200:**
- **Routes matched : 12.0.0.0**



```
AS50#show ip bgp
   Network              Path
*> 5.0.0.0              i
*> 1.0.0.0              100 i
*> 2.0.0.0              100 200 i
*> 3.0.0.0              300 i
*> 4.0.0.0              300 400 i
*> 10.0.0.0             300 400 1000 I

AS50#show ip bpg regexp 100
```

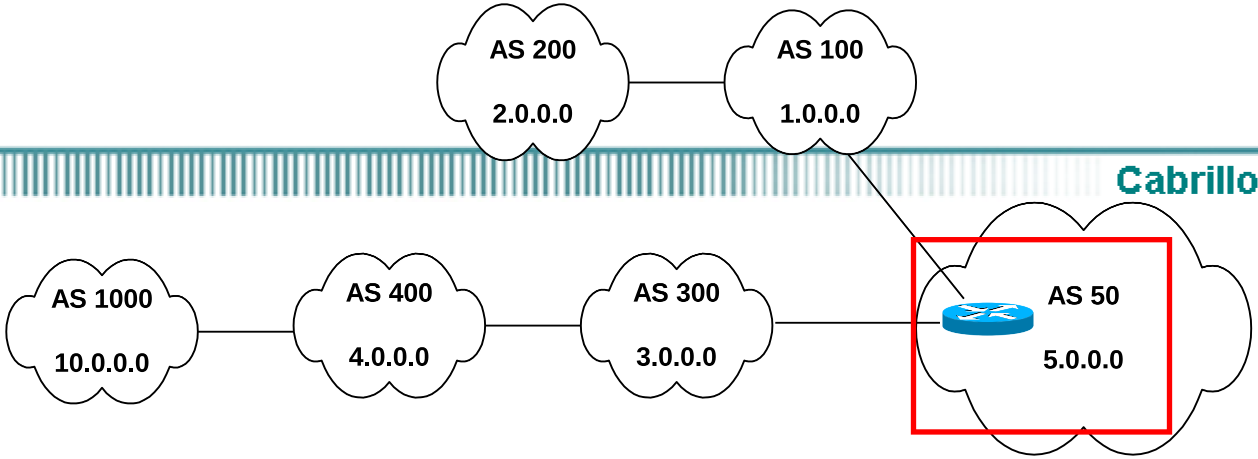
- **Match input string, AS_PATH, containing 100, including 1000**
- **Routes matched : 1.0.0.0, 2.0.0.0, 10.0.0.0**



```
AS50#show ip bgp
   Network                Path
*> 5.0.0.0                i
*> 1.0.0.0                100 i
*> 2.0.0.0                100 200 i
*> 3.0.0.0                300 i
*> 4.0.0.0                300 400 i
*> 10.0.0.0               300 400 1000 I
```

```
AS50#show ip bpg regexp ^100_
```

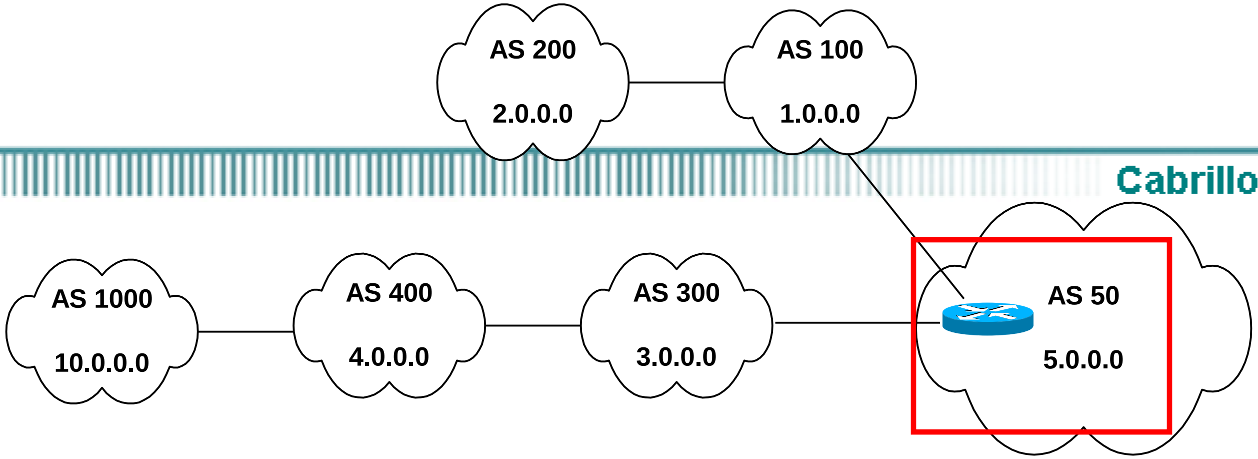
- **Match beginning of input string, AS_PATH, = 100**
- **Last prepended AS was 100:**
- **Routes matched : 1.0.0.0, 2.0.0.0**



```
AS50#show ip bgp
   Network              Path
*> 5.0.0.0              i
*> 1.0.0.0              100 i
*> 2.0.0.0              100 200 i
*> 3.0.0.0              300 i
*> 4.0.0.0              300 400 i
*> 10.0.0.0             300 400 1000 I

AS50# show ip bgp regexp _400$
```

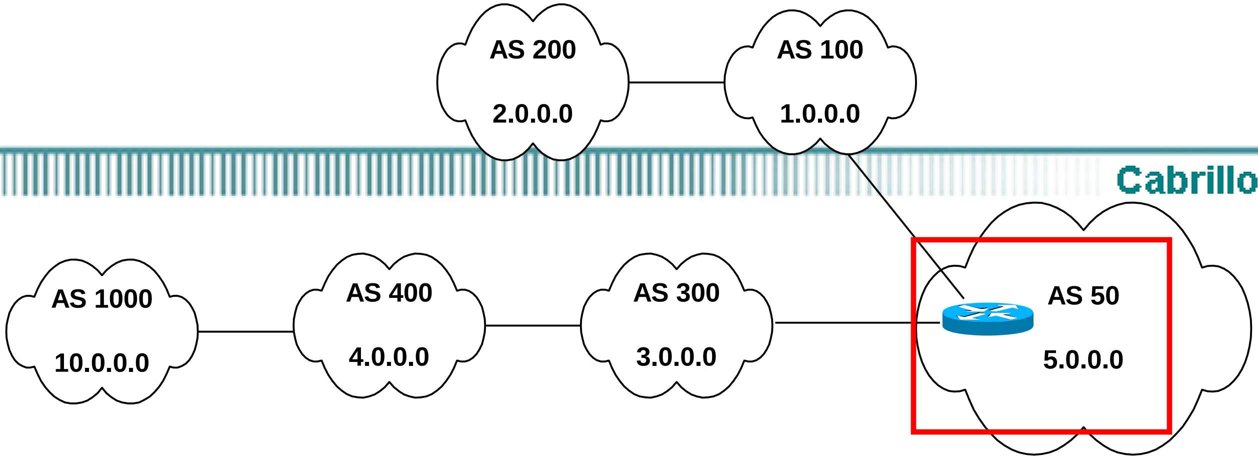
- **Match end of input string, AS_PATH, = 400**
- **Originating AS = 400:**
- **Routes matched : 4.0.0.0**



```
AS50#show ip bgp
   Network             Path
*> 5.0.0.0             i
*> 1.0.0.0             100 i
*> 2.0.0.0             100 200 i
*> 3.0.0.0             300 i
*> 4.0.0.0             300 400 i
*> 10.0.0.0            300 400 1000 I

AS50#show ip bpg regexp _400_
```

- **Match anywhere in input string, AS_PATH, 400**
- **Routes matched : 4.0.0.0, 10.0.0.0**

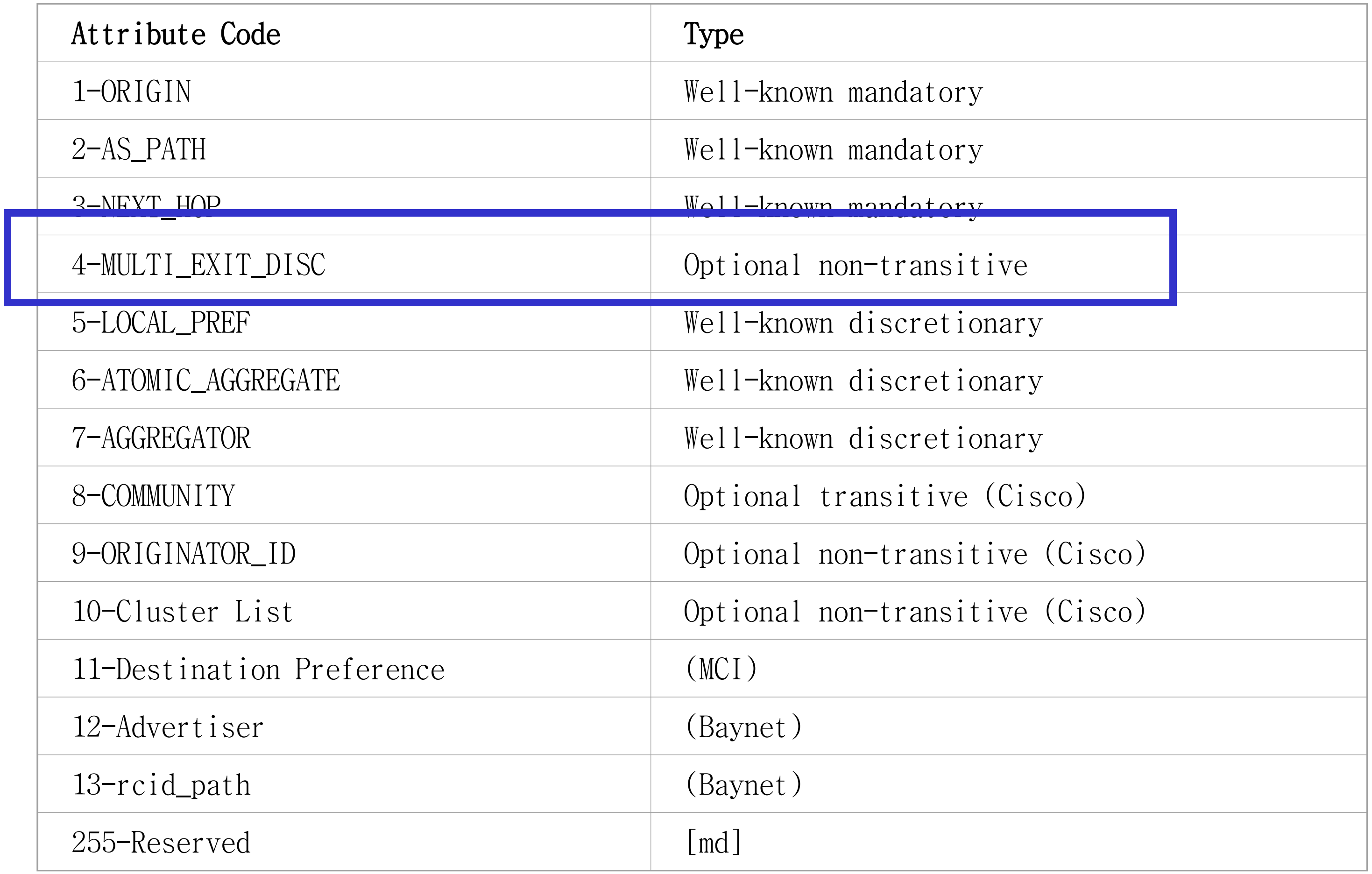


```
AS50#show ip bgp
   Network             Path
*> 5.0.0.0             i
*> 1.0.0.0             100 i
*> 2.0.0.0             100 200 i
*> 3.0.0.0             300 i
*> 4.0.0.0             300 400 i
*> 10.0.0.0            300 400 1000 I

AS50#show ip bgp regexp ^300$
```

- **Match input string that starts and ends at 300**
- **Routes that originated from directly connected AS 300 customer**
- **Routes matched : 3.0.0.0**

Path Attributes



Attribute Code	Type
1-ORIGIN	Well-known mandatory
2-AS_PATH	Well-known mandatory
3-NEXT_HOP	Well-known mandatory
4-MULTI_EXIT_DISC	Optional non-transitive
5-LOCAL_PREF	Well-known discretionary
6-ATOMIC_AGGREGATE	Well-known discretionary
7-AGGREGATOR	Well-known discretionary
8-COMMUNITY	Optional transitive (Cisco)
9-ORIGINATOR_ID	Optional non-transitive (Cisco)
10-Cluster List	Optional non-transitive (Cisco)
11-Destination Preference	(MCI)
12-Advertiser	(Baynet)
13-rcid_path	(Baynet)
255-Reserved	[md]

The MED attribute

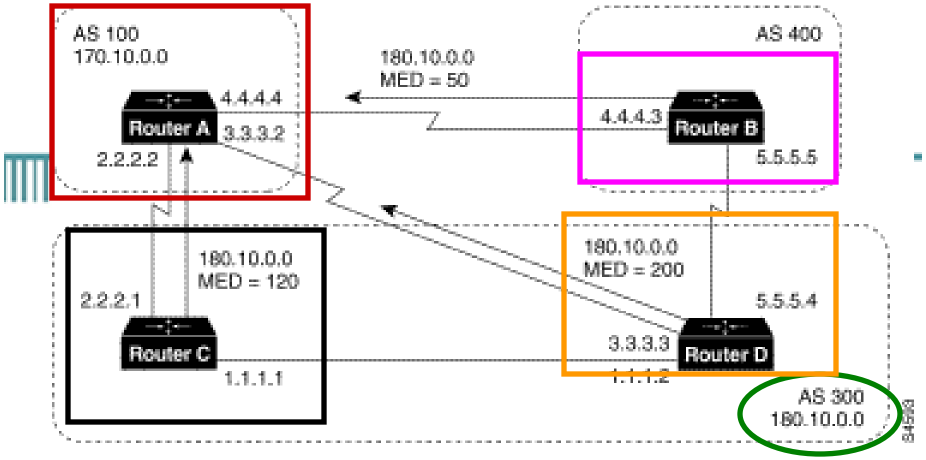
- The MULTI_EXIT_DISC (Multi-Exit Discriminator) attribute is an optional non-transitive attribute (type code 4).
- **Informs external neighbors about the preferred path** into an AS that has multiple entry points.
- A lower MULTI_EXIT_DISC (or MED) is preferred over a higher MED.

The MED attribute

Multi-Exit Discriminator Attribute

- The multi-exit discriminator (MED) attribute is a ***hint* to external neighbors** about the preferred path **into an AS** when there are **multiple entry points** into the AS.
- A **lower MED value is preferred** over a higher MED value.
- The **default** value of the **MED** attribute is **0**.
- Unlike local preference, the **MED attribute is exchanged between AS's**, but a **MED attribute that comes into an AS does not leave the AS.**
- When an update enters the AS with a certain MED value, that value is used for decision making within the AS.
- When BGP sends that update to another AS, the MED is reset to 0.
- Unless otherwise specified, the router compares MED attributes for paths from external neighbors that are in the same AS.
- **If you want MED attributes from neighbors in other ASes to be compared**, you must configure the **bgp always-compare-med** command.

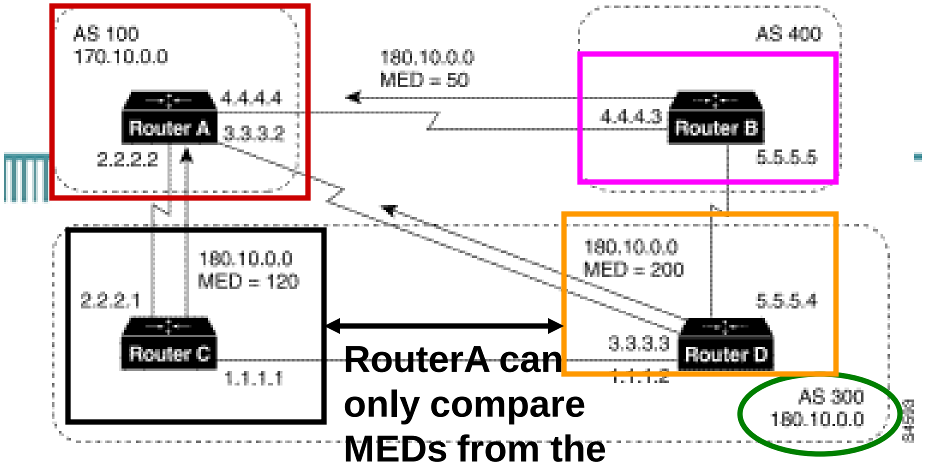
- **AS 100** receives updates regarding network **180.10.0.0** from Routers B, **C**, and D.
- Routers C and D are in AS 300, and Router B is in AS 400.



```
Router A
router bgp 100
   neighbor 2.2.2.1 remote-as 300
   neighbor 3.3.3.3 remote-as 300
   neighbor 4.4.4.3 remote-as 400
Router B
router bgp 400
   neighbor 4.4.4.4 remote-as 100
   neighbor 4.4.4.4 route-map
          SETMEDOUT out
   neighbor 5.5.5.4 remote-as 300
route-map SETMEDOUT permit 10
   set metric 50
```

```
Router C
router bgp 300
   neighbor 2.2.2.2 remote-as 100
   neighbor 2.2.2.2 route-map SETMEDOUT out
   neighbor 5.5.5.5 remote-as 400
   neighbor 1.1.1.2 remote-as 300
route-map SETMEDOUT permit 10
   set metric 120
Router D
router bgp 300
   neighbor 3.3.3.2 remote-as 100
   neighbor 3.3.3.2 route map SETMEDOUT out
   neighbor 1.1.1.1 remote-as 300
route-map SETMEDOUT permit 10
   set metric 200
```

- By **default**, BGP compares the MED attributes of routes coming from neighbors in the same external AS *as the route* (such as AS 300).
- **Router A** can only compare the MED attribute coming from **Router C (120)** to the MED attribute coming from **Router D (200)** even though the update coming from **Router B** has the lowest MED value.

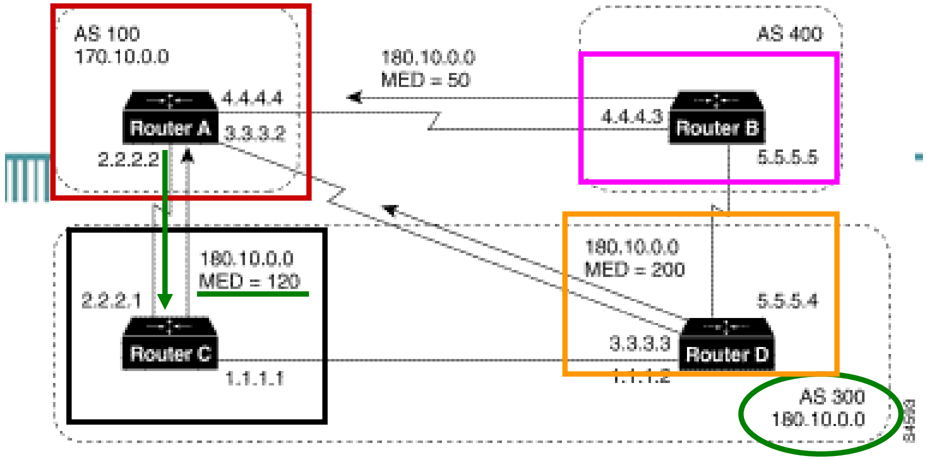


RouterA can only compare MEDs from the same AS

```
Router A
router bgp 100
   neighbor 2.2.2.1 remote-as 300
   neighbor 3.3.3.3 remote-as 300
   neighbor 4.4.4.3 remote-as 400
Router B
router bgp 400
   neighbor 4.4.4.4 remote-as 100
   neighbor 4.4.4.4 route-map
          SETMEDOUT out
   neighbor 5.5.5.4 remote-as 300
route-map SETMEDOUT permit 10
   set metric 50
```

```
Router C
router bgp 300
   neighbor 2.2.2.2 remote-as 100
   neighbor 2.2.2.2 route-map SETMEDOUT out
   neighbor 5.5.5.5 remote-as 400
   neighbor 1.1.1.2 remote-as 300
route-map SETMEDOUT permit 10
   set metric 120
Router D
router bgp 300
   neighbor 3.3.3.2 remote-as 100
   neighbor 3.3.3.2 route map SETMEDOUT out
   neighbor 1.1.1.1 remote-as 300
route-map SETMEDOUT permit 10
   set metric 200
```

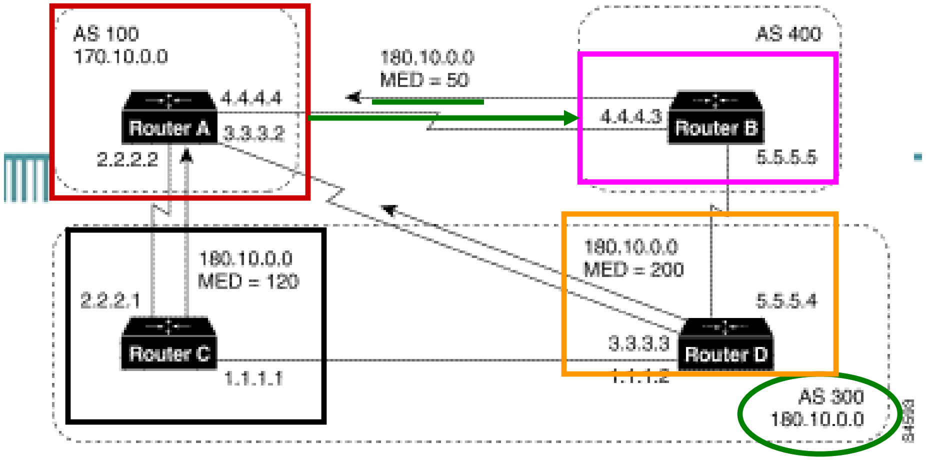
- **Router A will choose Router C** as the best path for reaching network **180.10.0.0**.



```
Router A
router bgp 100
   neighbor 2.2.2.1 remote-as 300
   neighbor 3.3.3.3 remote-as 300
   neighbor 4.4.4.3 remote-as 400
Router B
router bgp 400
   neighbor 4.4.4.4 remote-as 100
   neighbor 4.4.4.4 route-map
          SETMEDOUT out
   neighbor 5.5.5.4 remote-as 300
route-map SETMEDOUT permit 10
   set metric 50
```

```
Router C
router bgp 300
   neighbor 2.2.2.2 remote-as 100
   neighbor 2.2.2.2 route-map SETMEDOUT out
   neighbor 5.5.5.5 remote-as 400
   neighbor 1.1.1.2 remote-as 300
route-map SETMEDOUT permit 10
   set metric 120
Router D
router bgp 300
   neighbor 3.3.3.2 remote-as 100
   neighbor 3.3.3.2 route map SETMEDOUT out
   neighbor 1.1.1.1 remote-as 300
route-map SETMEDOUT permit 10
   set metric 200
```

- To force **Router A** to include updates for network **180.10.0.0** from **Router B** in the comparison, use the **bgp always-compare-med router** configuration command on **Router A**:
- **Router A will choose Router B** as the best next hop for reaching network **180.10.0.0** (assuming that all other attributes are the same).



Router A

```
router bgp 100
   neighbor 2.2.2.1 remote-as 300
   neighbor 3.3.3.3 remote-as 300
   neighbor 4.4.4.3 remote-as 400
   bgp always-compare-med
```

Router B

```
router bgp 400
   neighbor 4.4.4.4 remote-as 100
   neighbor 4.4.4.4 route-map
          SETMEDOUT out
   neighbor 5.5.5.4 remote-as 300
route-map SETMEDOUT permit 10
   set metric 50
```

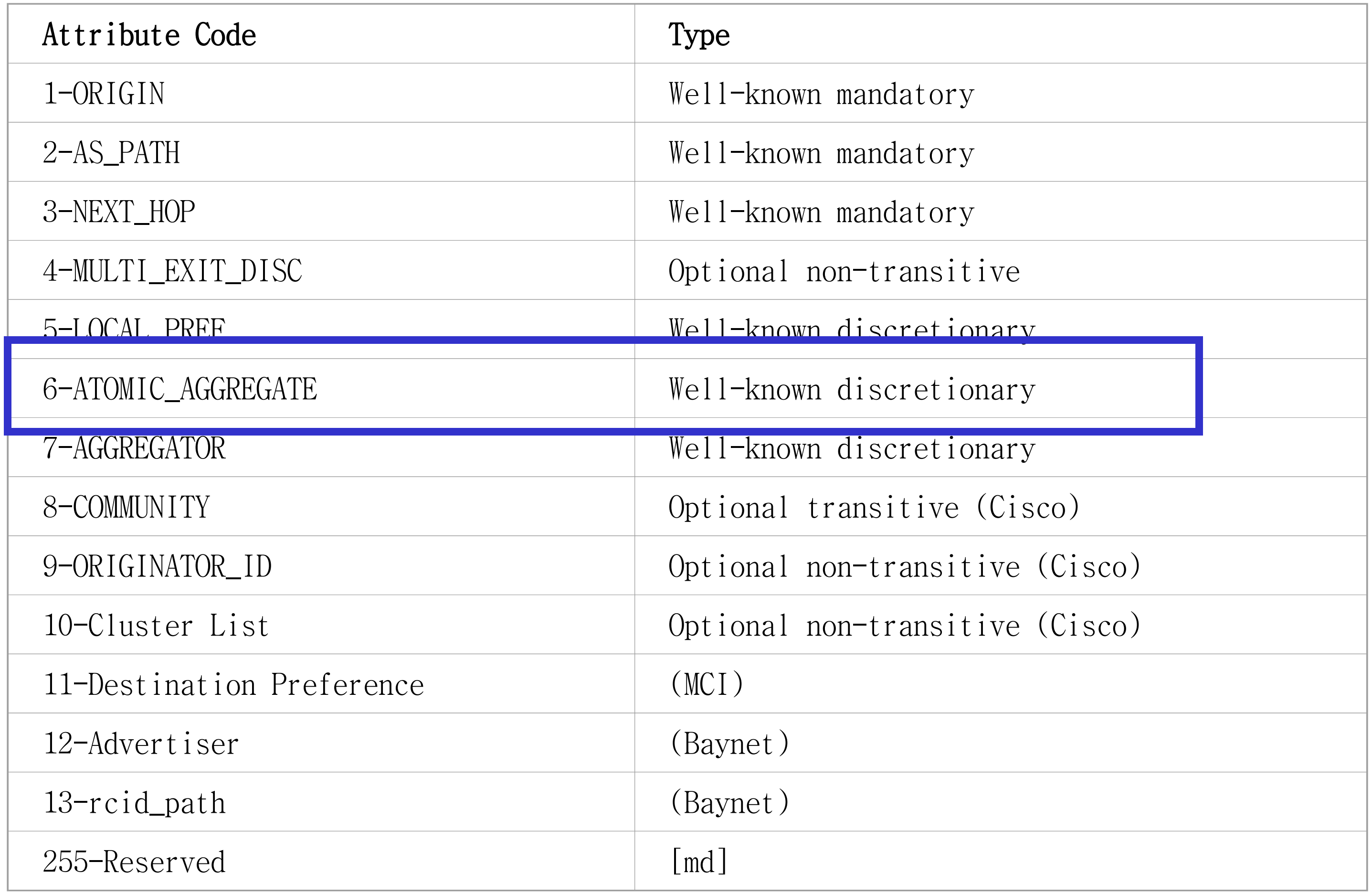
Router C

```
router bgp 300
   neighbor 2.2.2.2 remote-as 100
   neighbor 2.2.2.2 route-map SETMEDOUT out
   neighbor 5.5.5.5 remote-as 400
   neighbor 1.1.1.2 remote-as 300
route-map SETMEDOUT permit 10
   set metric 120
```

Router D

```
router bgp 300
   neighbor 3.3.3.2 remote-as 100
   neighbor 3.3.3.2 route map SETMEDOUT out
   neighbor 1.1.1.1 remote-as 300
route-map SETMEDOUT permit 10
   set metric 200
```

Path Attributes - FYI



Attribute Code	Type
1-ORIGIN	Well-known mandatory
2-AS_PATH	Well-known mandatory
3-NEXT_HOP	Well-known mandatory
4-MULTI_EXIT_DISC	Optional non-transitive
5-LOCAL_PREF	Well-known discretionary
6-ATOMIC_AGGREGATE	Well-known discretionary
7-AGGREGATOR	Well-known discretionary
8-COMMUNITY	Optional transitive (Cisco)
9-ORIGINATOR_ID	Optional non-transitive (Cisco)
10-Cluster List	Optional non-transitive (Cisco)
11-Destination Preference	(MCI)
12-Advertiser	(Baynet)
13-rcid_path	(Baynet)
255-Reserved	[md]

ATOMIC_AGGREGATE - FYI

- This attribute uses the **aggregate-address** command.
- A BGP speaking router can transmit overlapping routes to another BGP speaker.
- **Overlapping routes** are **non-identical routes that point to the same destination**.
- For example, 206.25.192.0/19 and 206.25.128.0/17 are overlapping, as the first route is included in the second route.
- The second route, 206.25.128.0/17, points to other more specific routes besides 206.25.192.0/19.
- When making a best path decision, a router always chooses the **more-specific path**.
- When advertising routes, however, the BGP speaker has several options with overlapping routes.

ATOMIC_AGGREGATE - FYI

Choices:

- Advertise both the more-specific and the less-specific route
- Advertise only the more-specific route
- Advertise only the non-overlapping part of the route
- Aggregate (summarize) the two routes and advertise the aggregate
- Advertise the less-specific route only
- Advertise neither route.

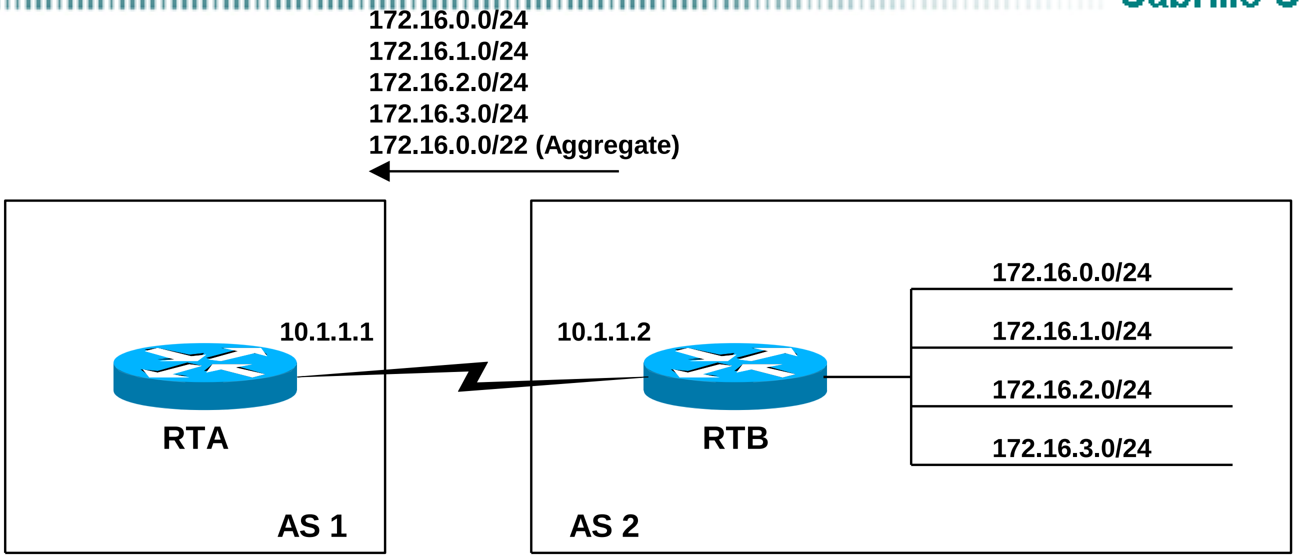
ATOMIC_AGGREGATE - FYI

- The **ATOMIC_AGGREGATE** is a well-know discretionary attribute (type code 6).
- The **ATOMIC_AGGREGATE** attribute is set to either “**True**” or “**False**.”
- If **true**, this attribute alerts BGP routers that multiple destinations have been grouped into a single update.
  - In other words, the **BGP router that sent the update had a more specific route to the destination, but did not send it.**
  - **ATOMIC_AGGREGATE** warns receiving routers that the information they are receiving is **not necessarily the most complete route information available.**

You can manually configure BGP to summarize routes by using the **aggregate-address** command, which has the following syntax:

```
Router(config-router)#aggregate-address address mask [as-
   set][summary-only] [suppress-map map-name][advertise-map map-
   name] [attribute-map map-name]
```

ATOMIC_AGGREGATE - FYI

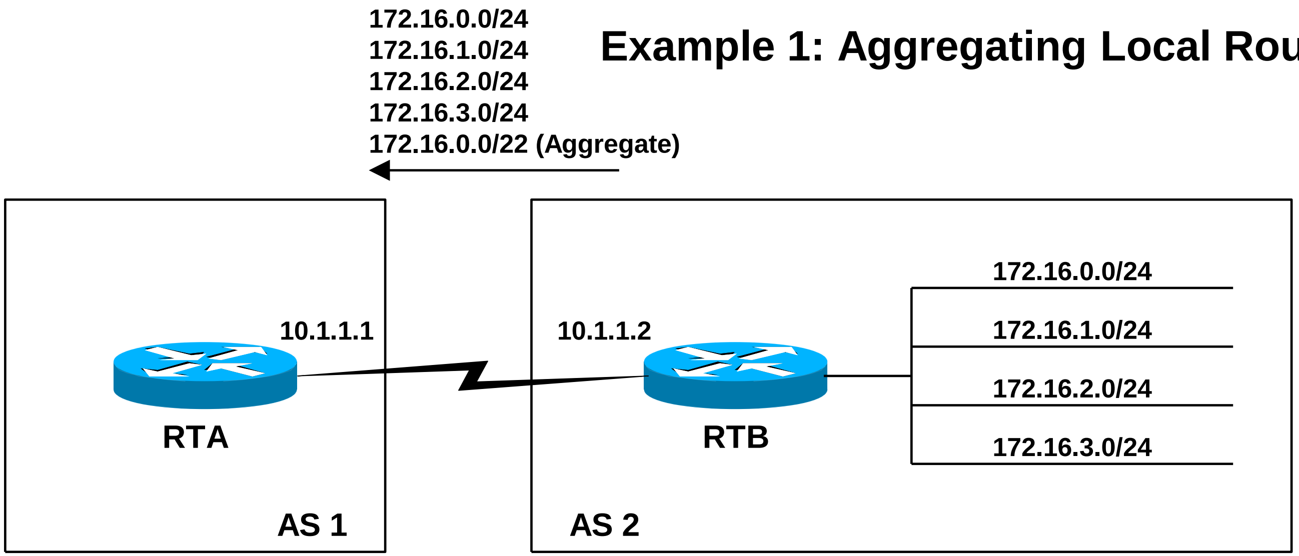


- The purpose of this command is to create an **aggregate** (summarized) entry in the BGP table.
- There are two ways to create an aggregate address under BGP:

1. Create a static entry in the routing table for the aggregate address and then advertise it with the network command.
2. Use the aggregate-address command.

- An aggregate is created only if a more-specific route to the aggregate exists in the BGP table.

Example 1: Aggregating Local Routes



RTA

```
router bgp 1
  neighbor 10.1.1.2 remote-as 2
```

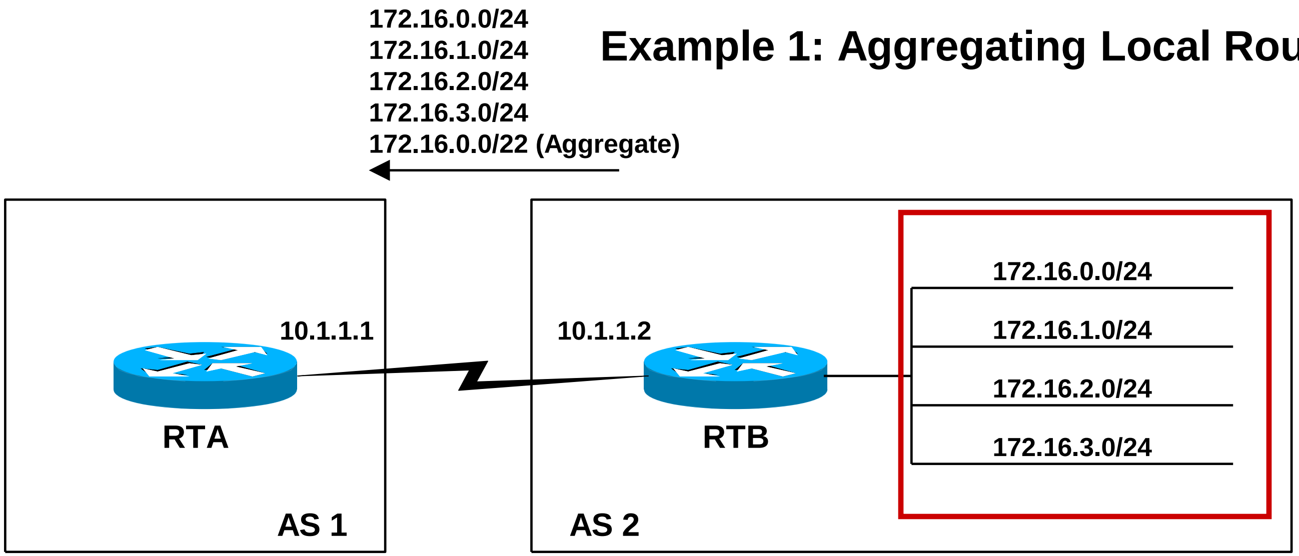
RTB

```
router bgp 2
  neighbor 10.1.1.1 remote-as 1
  network 172.16.0.0 mask {/24}
  network 172.16.1.0 mask {/24}
  network 172.16.2.0 mask {/24}
  network 172.16.3.0 mask {/24}
```

Before aggregating locally sourced routes, lets configure the more-specific networks.

- RTB has four loopbacks used to simulate the networks along with BGP network commands.
- RTA and RTB will have all 172.16.n.0/24 routes in its BGP table (show ip bgp)

Example 1: Aggregating Local Routes



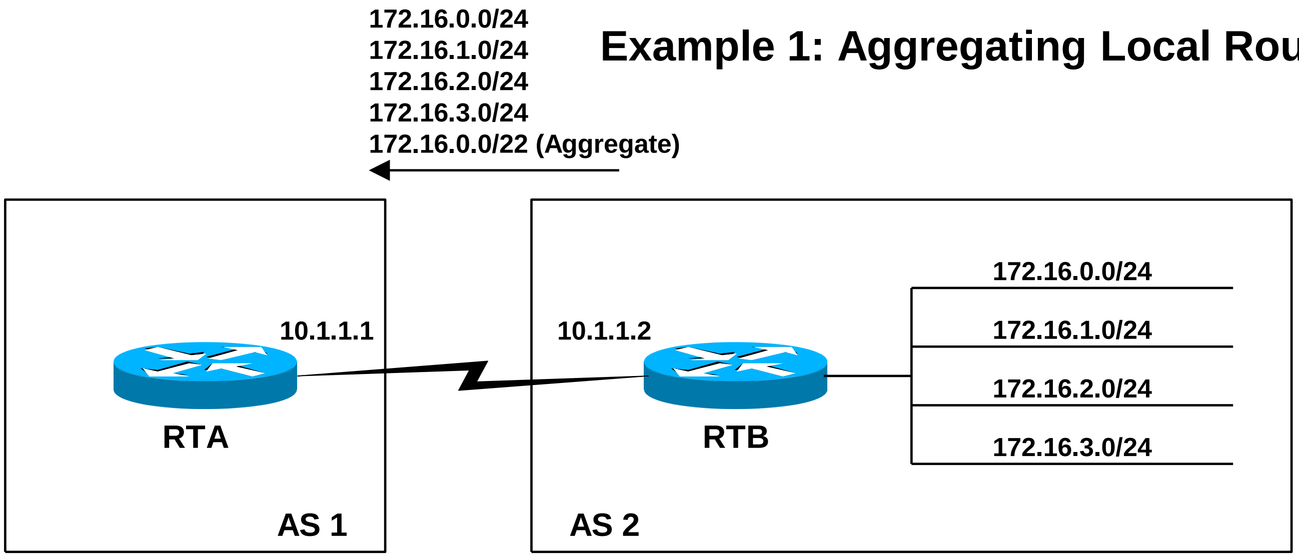
RTB

```
router bgp 2
  neighbor 10.1.1.1 remote-as 1
  network 172.16.0.0 mask {/24}
  network 172.16.1.0 mask {/24}
  network 172.16.2.0 mask {/24}
  network 172.16.3.0 mask {/24}
  aggregate-address 172.16.0.0
   255.255.252.0 {/22}
```

Now modify the BGP on RGB to enable the advertisement of the aggregate:

- We need only one of the more-specific network commands in RTB in order to send the aggregate, but by configuring all of them **the aggregate will be sent in case one of the networks goes down**.
- RTA and RTB will have all 172.16.n.0/22 routes in its BGP table (show ip bgp), **and** the the aggregate address of 172.16.0.0/22

Example 1: Aggregating Local Routes



show ip bpg 172.16.0.0 will display that this route has the “atomic-aggregate” attribute set.

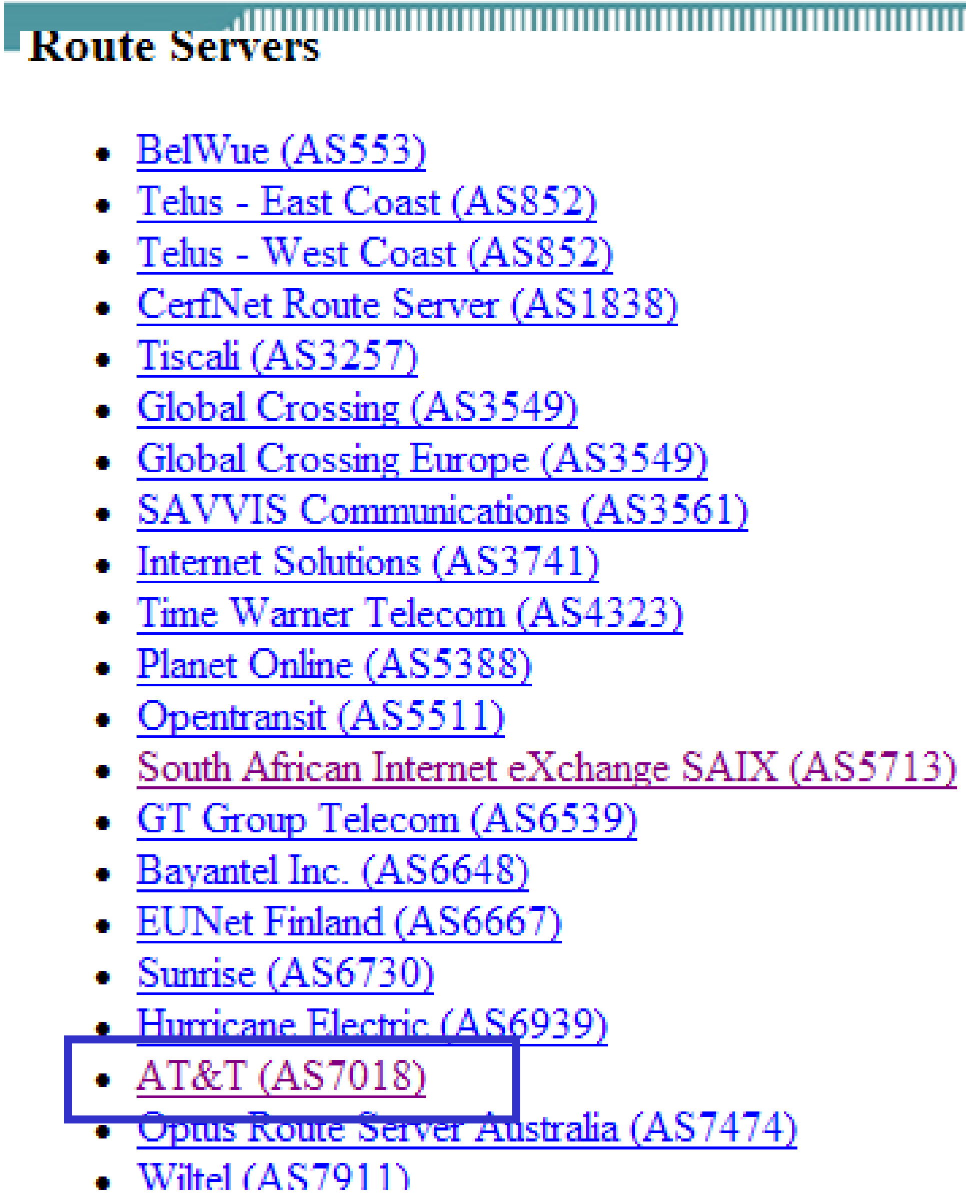
```
RTA#show ip bgp 172.16.0.0 255.255.252.0
BGP routing table entry for 172.16.0.0/22, version 18
Paths: (1 available, best #1)
<text omitted>
 Origin IGP, localpref 100, valid, external, atomic- aggregate, best
```

Summary of the BGP Path Selection Process

- BGP selects only one path as the best path.
- When the path is selected, BGP puts the selected path in its routing table and propagates the path to its neighbors.
- BGP uses the following criteria, in the order presented, to select a path for a destination:

1. If the path specifies a next hop that is inaccessible, drop the update.
2. Prefer the path with the largest weight.
3. If the weights are the same, prefer the path with the largest local preference.
4. If the local preferences are the same, prefer the path that was originated by BGP running on this router.
5. If no route was originated, prefer the route that has the shortest AS_path.
6. If all paths have the same AS_path length, prefer the path with the lowest origin type (where IGP is lower than EGP, and EGP is lower than Incomplete).
7. If the origin codes are the same, prefer the path with the lowest MED attribute.
8. If the paths have the same MED, prefer the external path over the internal path.
9. If the paths are still the same, prefer the path through the closest IGP neighbor.
10. Prefer the path with the lowest IP address, as specified by the BGP router ID.

Traceroute.org – www.traceroute.org



Route Servers

- BelWue (AS553)
- Telus - East Coast (AS852)
- Telus - West Coast (AS852)
- CerfNet Route Server (AS1838)
- Tiscali (AS3257)
- Global Crossing (AS3549)
- Global Crossing Europe (AS3549)
- SAVVIS Communications (AS3561)
- Internet Solutions (AS3741)
- Time Warner Telecom (AS4323)
- Planet Online (AS5388)
- Opentransit (AS5511)
- South African Internet eXchange SAIX (AS5713)
- GT Group Telecom (AS6539)
- Bayantel Inc. (AS6648)
- EUNet Finland (AS6667)
- Sunrise (AS6730)
- Hurricane Electric (AS6939)
- AT&T (AS7018)
- Optus Route Server Australia (AS7474)
- Wiltel (AS7911)

What is a Route Server?

http://www.inetdaemon.com/tools/route_servers.html

- A route server provides a look into the IP routing tables of the autonomous system in which the server resides. The concept of a route server has it's origins in the old Unix-based route serves that used to be located in the Network Access Points during the early days of the Internet. These Unix machines were configured with custom routing software ('routed', pronounced 'rout-dee'), designed specifically to make best-path calculations, and distribute a routing table to the routing devices forming the backbone of the Internet at these major peering points.
- As custom routing hardware became more and more powerful (and cheaper), most NAP and CIX managers started setting up Cisco routers with open logins. This reqires less manhours and less work than many other methods. You can telnet to these routers and get a direct look at another network's routing table, and test connectivity.

Cabrillo – 207.62.184.0

```
route-server>show ip route 207.62.184.0
Routing entry for 207.62.0.0/16, supernet
  Known via "bgp 65000", distance 20, metric 0
  Tag 7018, type external
  Last update from 12.123.1.236 1d12h ago
  Routing Descriptor Blocks:
  * 12.123.1.236, from 12.123.1.236, 1d12h ago
      Route metric is 0, traffic share count is 1
      AS Hops 3
      Route tag 7018
```

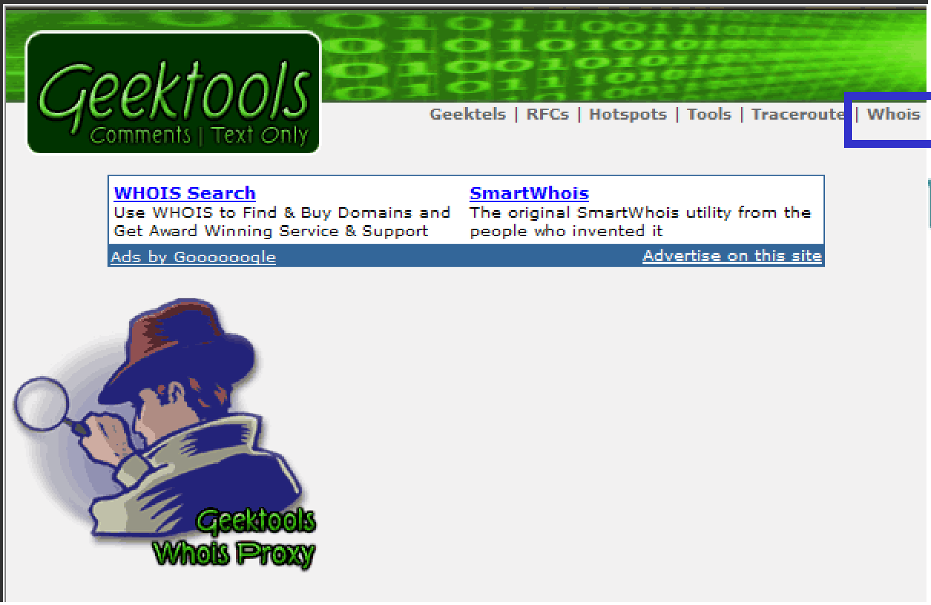
65000 – This AS

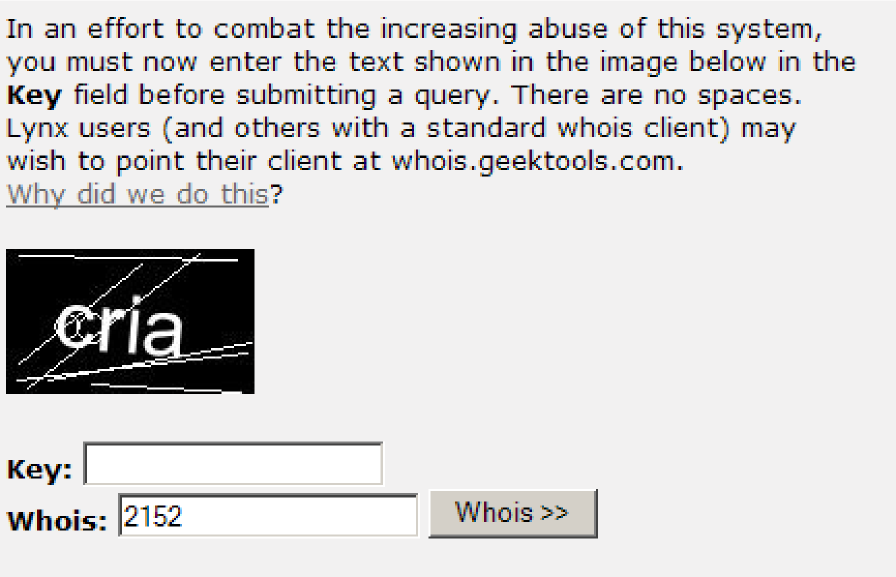
7018 – Next AS

```
route-server>show ip bgp 207.62.184.0
BGP routing table entry for 207.62.0.0/16, version 144563
Paths: (19 available, best #2, table Default-IP-Routing-Table)
  Not advertised to any peer
  7018 3356 2152, (received-only)
    12.0.1.1 from 12.0.1.63 (12.0.1.63)
      Origin IGP, localpref 100, valid, external
      Community: 7018:5000
  7018 3356 2152, (received & used)
    12.123.1.236 from 12.123.1.236 (12.123.1.236)
      Origin IGP, localpref 100, valid, external, best
      Community: 7018:5000
  7018 3356 2152, (received & used)
    12.123.13.241 from 12.123.13.241 (12.123.13.241)
      Origin IGP, localpref 100, valid, external
      Community: 7018:5000
    More…
```

- ASNumber: 7018
  ASName: ATT-INTERNET4
- ASNumber: 3356
  ASName: LEVEL3
- ASNumber: 2152
  ASName: CSUNET-NW

Geektools.com

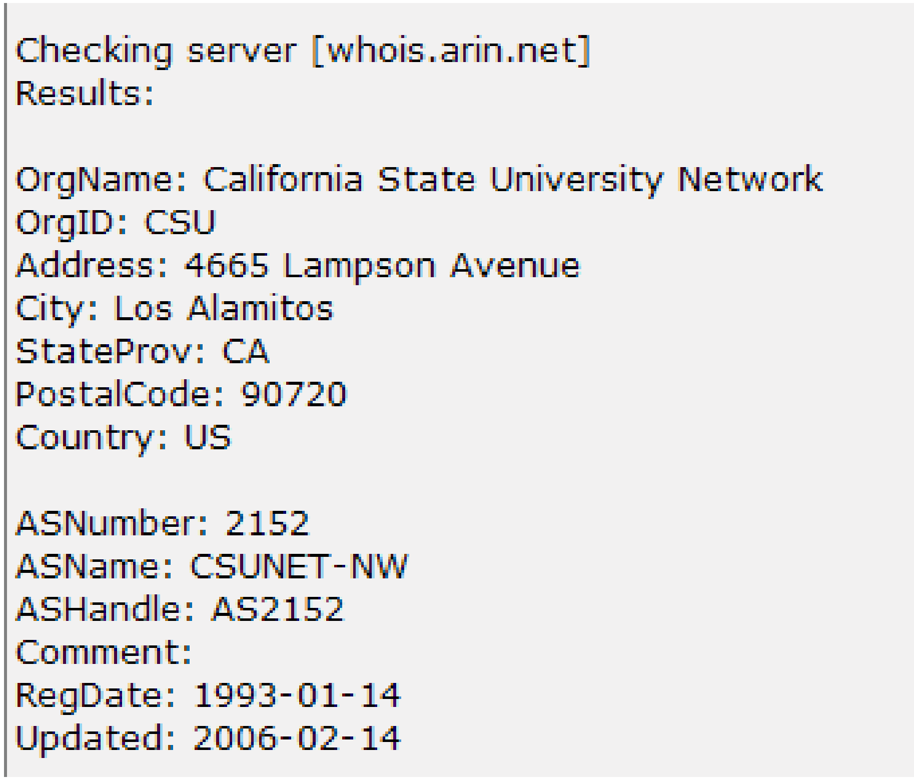




In an effort to combat the increasing abuse of this system, you must now enter the text shown in the image below in the **Key** field before submitting a query. There are no spaces. Lynx users (and others with a standard whois client) may wish to point their client at whois.geektools.com.
Why did we do this?

Key:

Whois: 2152



```
Checking server [whois.arin.net]
Results:

OrgName: California State University Network
OrgID: CSU
Address: 4665 Lampson Avenue
City: Los Alamitos
StateProv: CA
PostalCode: 90720
Country: US

ASNumber: 2152
ASName: CSUNET-NW
ASHandle: AS2152
Comment:
RegDate: 1993-01-14
Updated: 2006-02-14
```

Using regular expressions

```
route-server>show ip bgp regexp _2152_
BGP table version is 430053, local router ID is
   10.1.2.5
Status codes: s suppressed, d damped, h history, *
   valid, > best, i - internal, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 44.0.0.0         12.123.1.236                           0 7018 22822 22822 2152 7377 i
*                   12.123.17.244                          0 7018 22822 22822 2152 7377 i
*                   12.123.13.241                          0 7018 22822 22822 2152 7377 i
*                   12.123.21.243                          0 7018 22822 22822 2152 7377 i
*                   12.123.45.252                          0 7018 22822 22822 2152 7377 i
*                   12.123.139.124                         0 7018 22822 22822 2152 7377 i
*                   12.123.9.241                           0 7018 22822 22822 2152 7377 i
*                   12.123.137.124                         0 7018 22822 22822 2152 7377 i
*                   12.123.142.124                         0 7018 22822 22822 2152 7377 i
 --More--
```

```
route-server.exodus.net>show ip bgp regexp 2150$
<continued>  Includes Cabrillo       Best Route

   *> 207.62.0.0/16     12.123.1.236          0 7018 3356 2152 i
   *                    12.123.13.241         0 7018 3356 2152 i
   *                    12.123.21.243         0 7018 3356 2152 i
   *                    12.123.17.244         0 7018 3356 2152 i
   *                    12.123.9.241          0 7018 3356 2152 i
   *                    12.123.142.124        0 7018 3356 2152 i
   *                    12.123.139.124        0 7018 3356 2152 i
   *                    12.123.45.252         0 7018 3356 2152 i
   *                    12.123.25.245         0 7018 3356 2152 I
   More…

route-server>show ip route 207.62.184.0
Routing entry for 207.62.0.0/16, supernet
  Known via "bgp 65000", distance 20, metric 0
  Tag 7018, type external
  Last update from 12.123.1.236 1d12h ago
  Routing Descriptor Blocks:
  * 12.123.1.236, from 12.123.1.236, 1d12h ago
      Route metric is 0, traffic share count is 1
      AS Hops 3
      Route tag 7018
```

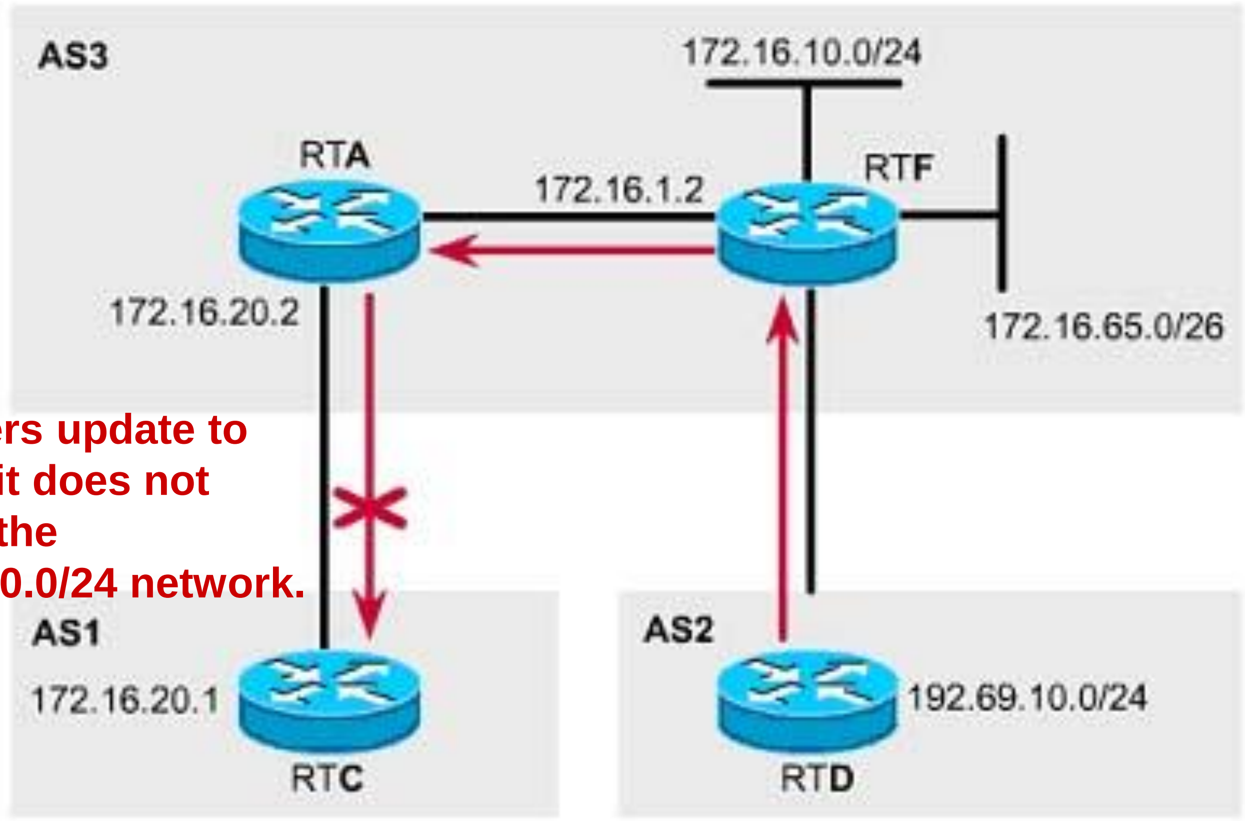
FYI

- The rest of the presentation is FYI

BGP Route Filtering

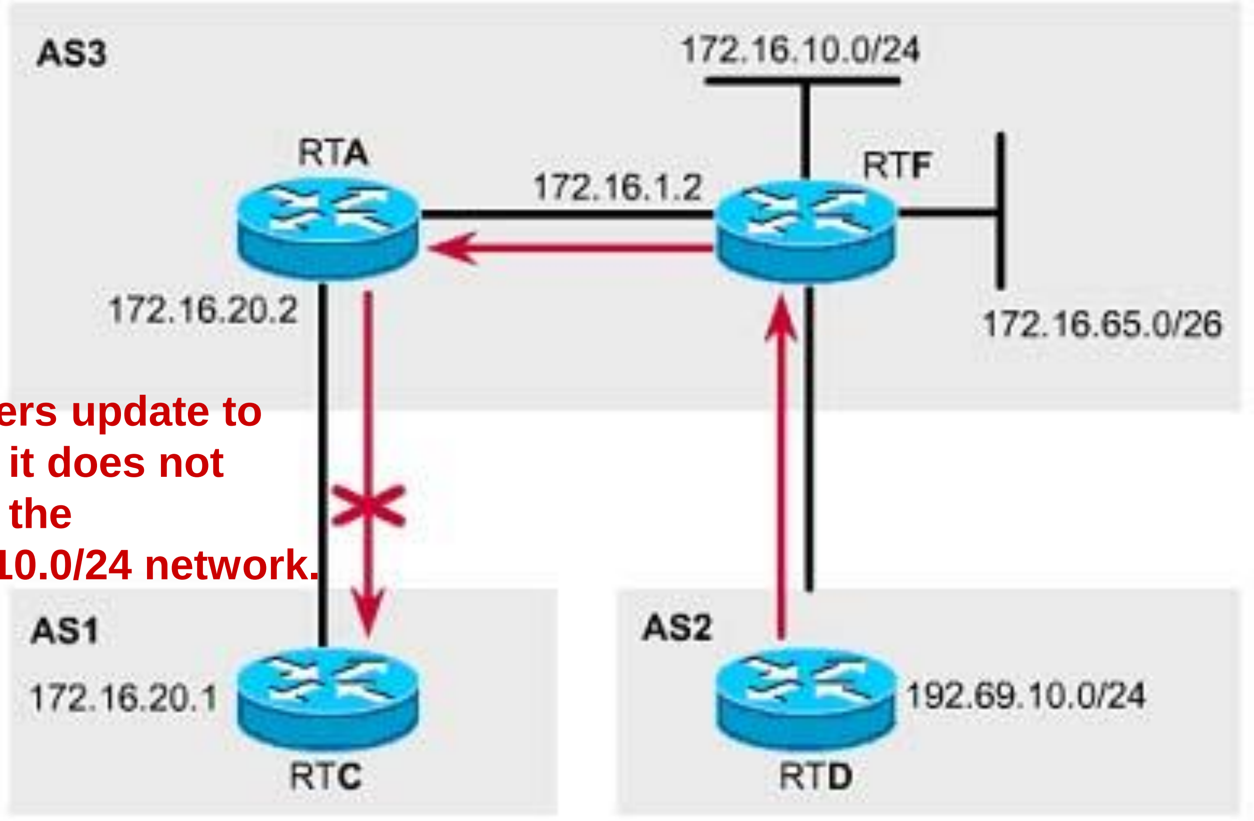
- Route filtering empowers a BGP speaker to **choose what routes to exchange with any of its BGP peers.**
- Route filtering is the cornerstone of policy routing.
  - An AS can identify inbound traffic it is willing to accept by filtering its outbound advertisements
  - An AS can control what routes its outbound traffic uses by specifying the routes to accept from EBGP neighbors
- Even more precise policies can be defined via route filters.
- For example, **BGP routes passing through a filter can have their attributes manipulated to affect the best-path decision process**.
- You can apply route filters to or from a particular neighbor by using the **distribute-list** command.

BGP Route Filtering



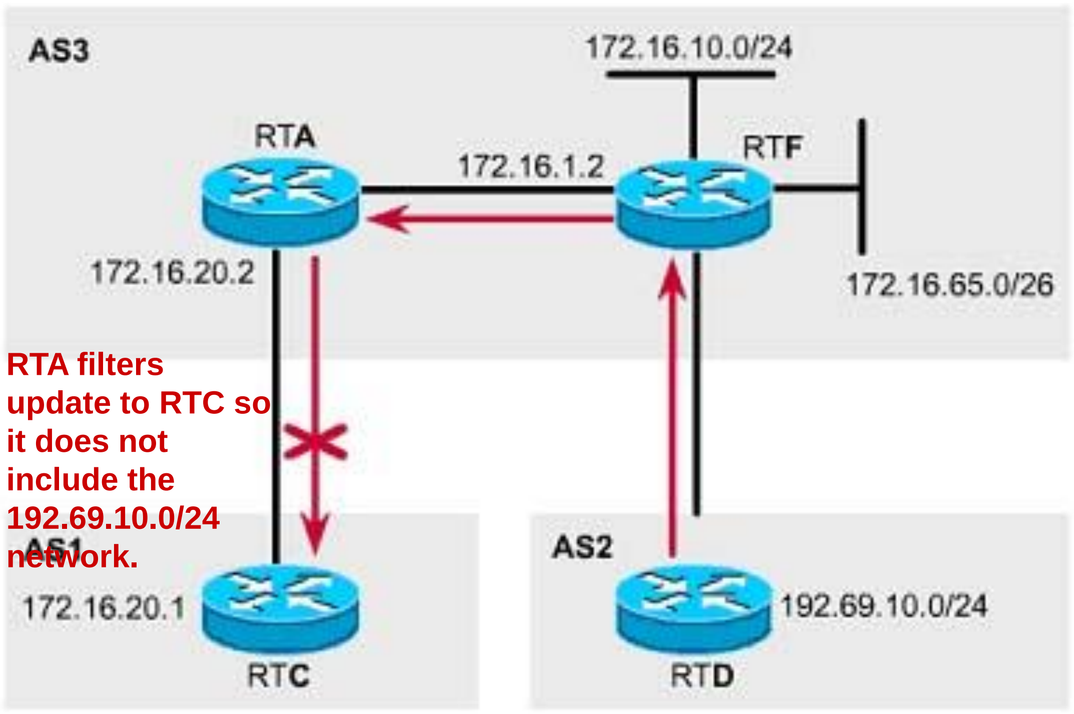
RTA filters update to RTC so it does not include the 192.69.10.0/24 network.

The distribute-list command can be used to filter updates so that AS1 does not receive transit traffic to network 192.69.10.0 /24.



RTA filters update to RTC so it does not include the 192.69.10.0/24 network.

```
RTA(config)#router bgp 3
RTA(config-router)#neighbor 172.16.1.2 remote-as 3
RTA(config-router)#neighbor 172.16.20.1 remote-as 1
RTA(config-router)#neighbor 172.16.20.1 distribute-list 1 out
RTA(config-router)#exit
RTA(config)#access-list 1 deny 192.69.10.0 0.0.0.255
RTA(config)#access-list 1 permit any
```



RTA filters update to RTC so it does not include the 192.69.10.0/24 network.

- The **distribute-list** keyword, used as part of a BGP **neighbor** statement, prevents RTA from advertising prefix 192.69.10.0/24 to RTC.
- The access list is used to identify the prefixes to be filtered, and the **distribute-list** and **out** keywords apply the filter to outgoing updates.
- Whereas configuring BGP **neighbor** statements to include the **distribute-list** keyword is effective for filtering specific routes, controlling supernets can be a bit trickier.

```
RTA(config)#router bgp 3
RTA(config-router)#neighbor 172.16.1.2 remote-as 3
RTA(config-router)#neighbor 172.16.20.1 remote-as 1
RTA(config-router)#neighbor 172.16.20.1 distribute-list 1 out
RTA(config-router)#exit
RTA(config)#access-list 1 deny 192.69.10.0 0.0.0.255
RTA(config)#access-list 1 permit any
```

BGP Route Filtering

- Configuring a distribute list relies on creating an access list.
- If we use a **standard access list**, we are afforded only limited functionality.
- **What if you want to advertise an aggregate address of 172.16.0.0 /16, but not the individual subnets themselves?**
- **A standard access list would not work** because it permits more than is desired, since it filters based on the network address only.
- For example, this access list would permit not only the 172.16.0.0/16 summary, but also all the components of that summary as well:

```
access-list 1 permit 172.16.0.0 0.0.255.255
```

- To restrict the update to the 172.16.0.0/16 summary, you can use an **extended access list**.
- In the case of a BGP route filter, an extended list matches,
  - first, the network address,
  - second, the subnet mask of the prefix.
- Both **network** and **mask** are paired with their own wildcard bitmask, using the following syntax:

```
Router(config)#access-list number permit|deny network
   network-wildcard mask mask-wildcard
```

- Using this configuration, RTA would not send a subnet route (such as 172.16.0.0 /17 or 172.16.10.0 /24) in an update to AS1.

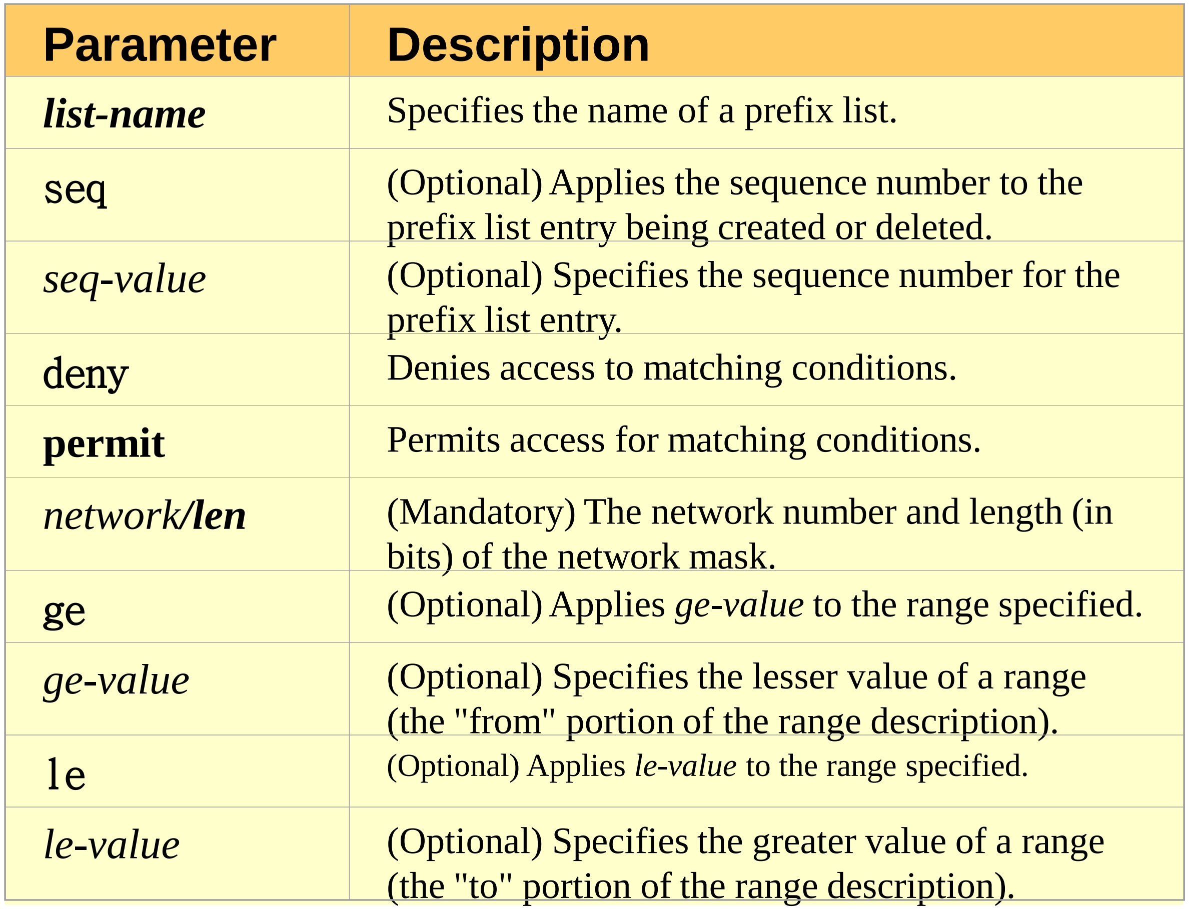
```
RTA(config)#router bgp 3
RTA(config-router)#neighbor 172.16.1.2 remote-as 3
RTA(config-router)#neighbor 172.16.20.1 remote-as 1
RTA(config-router)#neighbor 172.16.20.1 distribute-list 101 out
RTA(config-router)#exit
RTA(config)#access-list 101 permit ip 172.16.0.0 0.0.255.255
   255.255.0.0 0.0.0.0
```

BGP Route Filtering - Prefix lists

- If using an extended access list to accomplish this type of filtering seems confusing to you, you are not alone.
- **Improved user-friendliness** was one of the factors that motivated Cisco to include the **ip prefix-list** command in IOS 12.0.
- You can use prefix lists as an alternative to access lists with many BGP route-filtering commands.
- You must define a prefix list before you can apply it as a route filter.
- The Cisco IOS allows a very flexible configuration procedure, where each statement can be assigned its own sequence numbers.
- There is an **implicit deny at the end of each prefix list**.
- To define a prefix list, use the **ip prefix-list command**, which has the following syntax:

```
Router(config)#ip prefix-list list-name [seq seq-
   value] deny|permit network/len [ge ge-value] [le le-
   value]
```

BGP Route Filtering - Prefix lists



Parameter	Description
list-name	Specifies the name of a prefix list.
seq	(Optional) Applies the sequence number to the prefix list entry being created or deleted.
seq-value	(Optional) Specifies the sequence number for the prefix list entry.
deny	Denies access to matching conditions.
permit	Permits access for matching conditions.
*network/**len***	(Mandatory) The network number and length (in bits) of the network mask.
ge	(Optional) Applies *ge-value* to the range specified.
ge-value	(Optional) Specifies the lesser value of a range (the "from" portion of the range description).
le	(Optional) Applies *le-value* to the range specified.
le-value	(Optional) Specifies the greater value of a range (the "to" portion of the range description).

Example:

```
RTA(config)#ip prefix-list ELMO permit 172.16.0.0/16
RTA(config)#router bgp 100
RTA(config-router)#neighbor 192.168.1.1 remote-as 200
RTA(config-router)#neighbor 192.168.1.1 prefix-list ELMO out
```

- Restricts the update to the 172.16.0.0/16 summary
- Using this configuration, RTA would not send a subnet route (such as 172.16.0.0 /17 or 172.16.10.0 /24) in an update to AS1.

BGP Route Filtering - Prefix lists

- The real power of the **ip prefix-list** command is in its optional parameters.
- The keywords **ge** and **le** can be used to specify the range of the **prefix length** to be matched for prefixes that are more specific than the *network/len* value.
- The prefix-length range is assumed to be from *ge-value* to 32 if only the **ge** attribute is specified, and from *len* to *le-value* if only the **le** attribute is specified.
- For example, to accept a mask length of up to 24 bits in **routes** with the prefix 192.0.0.0/8, (ie.192.1.0.0/16, 192.2.10.0/24) and deny more specific routes (192.168.10.128/25), use the commands as shown in.

```
RTA(config)#ip prefix-list GROVER permit 192.0.0.0/8 le 24
RTA(config)#ip prefix-list GROVER deny 192.0.0.0/8 ge 25
```

BGP Route Filtering - Prefix lists

- The **le** and **ge** keywords can be used together, in the same statement:

```
RTA(config)#ip prefix-list OSCAR permit 10.0.0.0/8 ge
   16 le 24
```

- This list permits all prefixes in the 10.0.0.0/8 address space that have a mask of between 16 and 24 bits.

Examples - The following examples show how a prefix list can be used.

- To deny the default route 0.0.0.0/0:

  ```
  ip prefix-list abc deny 0.0.0.0/0
  ```

- To permit the prefix 35.0.0.0/8:

  ```
  ip prefix-list abc permit 35.0.0.0/8
  ```

The following examples show how to specify a group of prefixes.

- To accept a mask length of up to 24 bits in routes with the prefix 192/8:

  ```
  ip prefix-list abc permit 192.0.0.0/8 le 24
  ```

- To deny mask lengths greater than 25 bits in routes with a prefix of 192/8:

  ```
  ip prefix-list abc deny 192.0.0.0/8 ge 25
  ```

- To permit mask lengths from 8 to 24 bits in all address space:

  ```
  ip prefix-list abc permit 0.0.0.0/0 ge 8 le 24
  ```

- To deny mask lengths greater than 25 bits in all address space:

  ```
  ip prefix-list abc deny 0.0.0.0/0 ge 25
  ```

BGP Route Filtering - Prefix lists

- Each prefix list entry is assigned a sequence number, either by default or manually by an administrator.
- By numbering the prefix list statements, new entries can be inserted at any point in the list, which is important because routers test for prefix list matches from lowest sequence number to highest.
- By default, the entries of a prefix-list will have sequence values of 5,10, 15, etc.
- To disable this: RTR(config)# no ip prefix-list sequence-number
- Sequence numbers can be created using the command:

```
Router(config)#ip prefix-list list-name [seq seq-value]
   deny|permit network/len [ge ge-value] [le le-value]
```

```
RTA#show ip prefix-list
ip prefix-list ELMO: 3 entries
   seq 5 deny 0.0.0.0/0
   seq 10 permit 172.16.0.0/16
   seq 15 permit 192.168.0.0/16 le 24
```

Default Routes

- It is important to control default information in BGP because improper configuration can cause serious Internet routing problems.
- For example, a misconfigured BGP speaker could end up flooding a default route to all of its neighbors and quickly find itself consumed with default routed traffic from surrounding autonomous systems.
- To protect against misadvertisements, the Cisco IOS provides a way to target default information at a specific neighbor by using the **default-originate** option with the **neighbor** command:

```
RTC(config)#router bgp 3
RTC(config-router)#neighbor 172.16.20.1 remote-as 1
RTC(config-router)#neighbor 172.16.20.1 default-
originate
```

- If RTC is configured as shown in this configuration, it will send default information only to the specified neighbor.

Default Routes

- If a BGP router is to be configured to advertise a default to all of its peers, use the **`network`** command shown as follows:
- **Note:** Both neighbors, 172.16.20.1 and 172.17.1.1, will receive a default route from RTC.

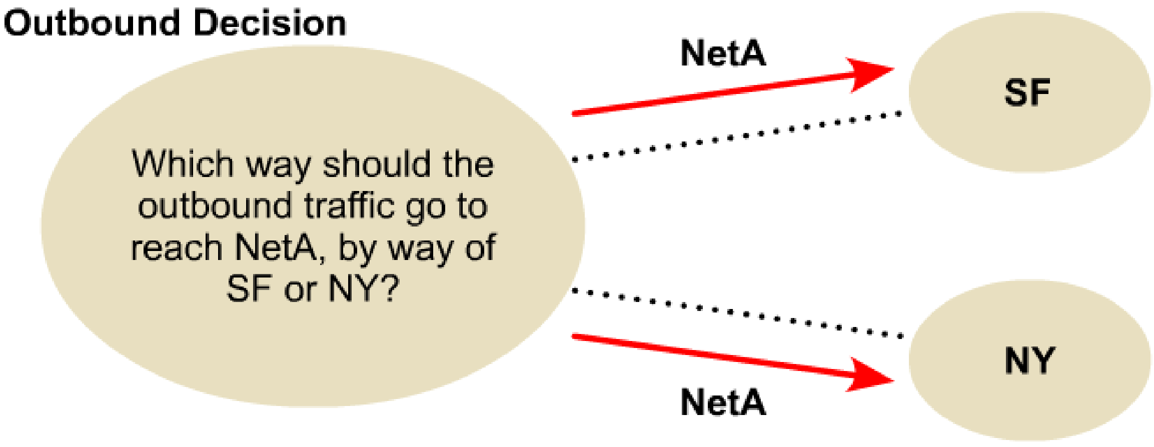
```
RTC(config)#router bgp 3
RTC(config-router)#neighbor 172.16.20.1 remote-as 1
RTC(config-router)#neighbor 172.17.1.1 remote-as 2
RTC(config-router)#network 0.0.0.0
```

- Many network administrators choose to filter dynamically learned default routes to avoid situations in which traffic ends up where it is not supposed to be.
- Without dynamically learned default routes, a router must be statically configured with default information.
- Statically configured default routes typically provide more control over routing within an AS.

Symmetry

- Symmetry is achieved when traffic leaving the AS from one exit point comes back through the same point.
- Symmetry always exists if an AS maintains a single connection to outside networks.
- However, the need for redundancy often results in multihoming an AS. If an AS has many different links to the outside world, traffic tends to flow asymmetrically.
- An asymmetrical traffic flow can result in increased delay and other routing problems.
- In general, customers and providers would like to see their traffic come back by way of the same point or close to the same point that it left the AS.
- To promote symmetry, choose a primary path and configure routing policies that force traffic to flow along this path.
- A default route with a low administrative distance or a high Local Preference might serve to control the flow of outbound traffic, but inbound traffic can be more complex to manipulate.
- Through appropriate planning and use of BGP attributes and route filters, an AS can control which paths the outside world finds most desirable.

Load Balancing


Outbound Decision
Which way should the outbound traffic go to reach NetA, by way of SF or NY?
NetA
SF
NY
NetA

- Load balancing is the capability to divide data traffic over multiple connections.
- A BGP speaker may learn two identical EBGP paths for a prefix from a neighboring AS.
- If this happens, it will choose the path with the lowest route ID as the best path.
- This best path is installed in the IP routing table.
- To enable BGP load balancing over equal cost paths, use the **`maximum-paths`** command, which has the following syntax:

```
Router(config-router)# maximum-paths number
```

- BGP supports a maximum of six paths per destination, but only if they are sourced from the same AS.
- By default, BGP will install only one path to the IP routing table.

Ch. 9 – BGP (Part 2)

Cabrillo College

CCNP 1 version 3.0

Rick Graziani

Cabrillo College